Introduction to HIPE

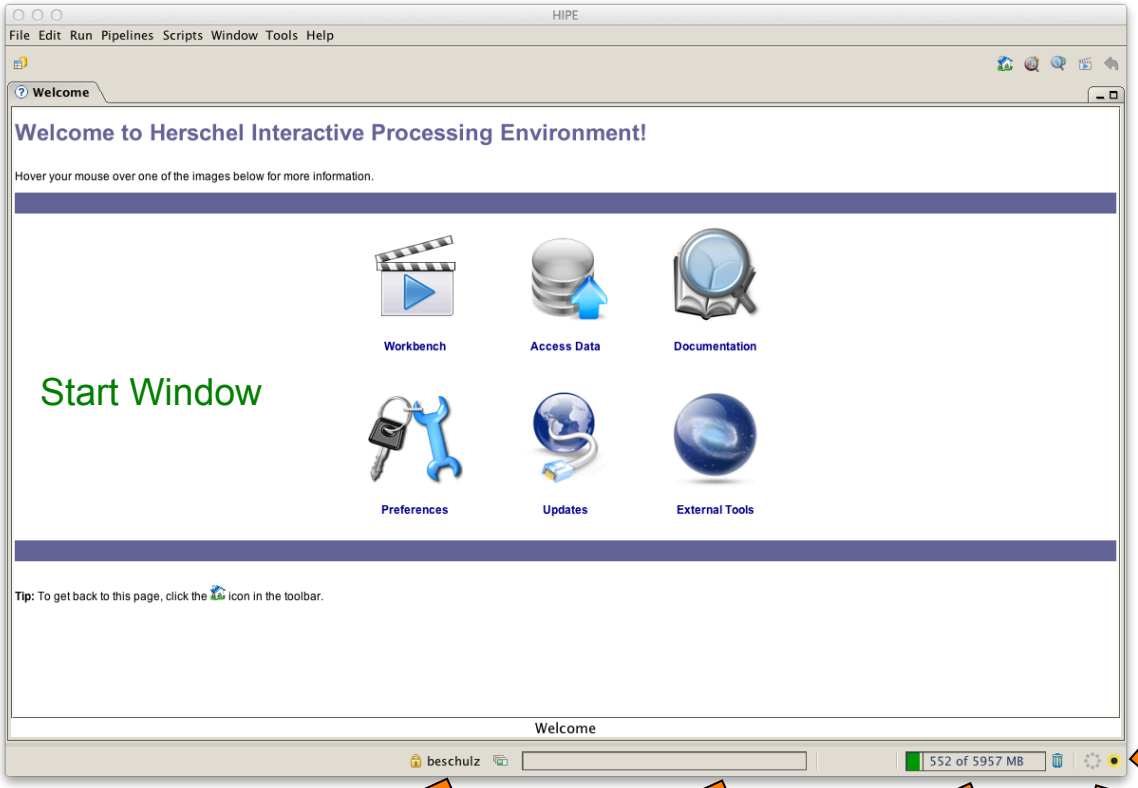
Bernhard Schulz
NHSC/IPAC

What is HIPE

- HIPE (Herschel Interactive Processing Environment)
  - GUI front end to the Herschel Common Science System (HCSS) for data processing.
- Written in Java
  - supported under Linux, Mac OS, MS Windows
- The software was developed in parallel to the mission hardware.
- All standard data products in the Herschel Archive were processed with this software.
- The code is free, open, and the same software is available to astronomers to repeat and improve Herschel data processing.
- HIPE was built to make the access to the data and data reduction procedures more user-friendly.
- Supports Jython as user scripting language.
- This presentation is here to whet your appetite and give you a basic introduction and overview of its capabilities.

First Look at HIPE

Warning: Closing the window will terminate the session immediately.



Start Window

SAMP-connected indicator

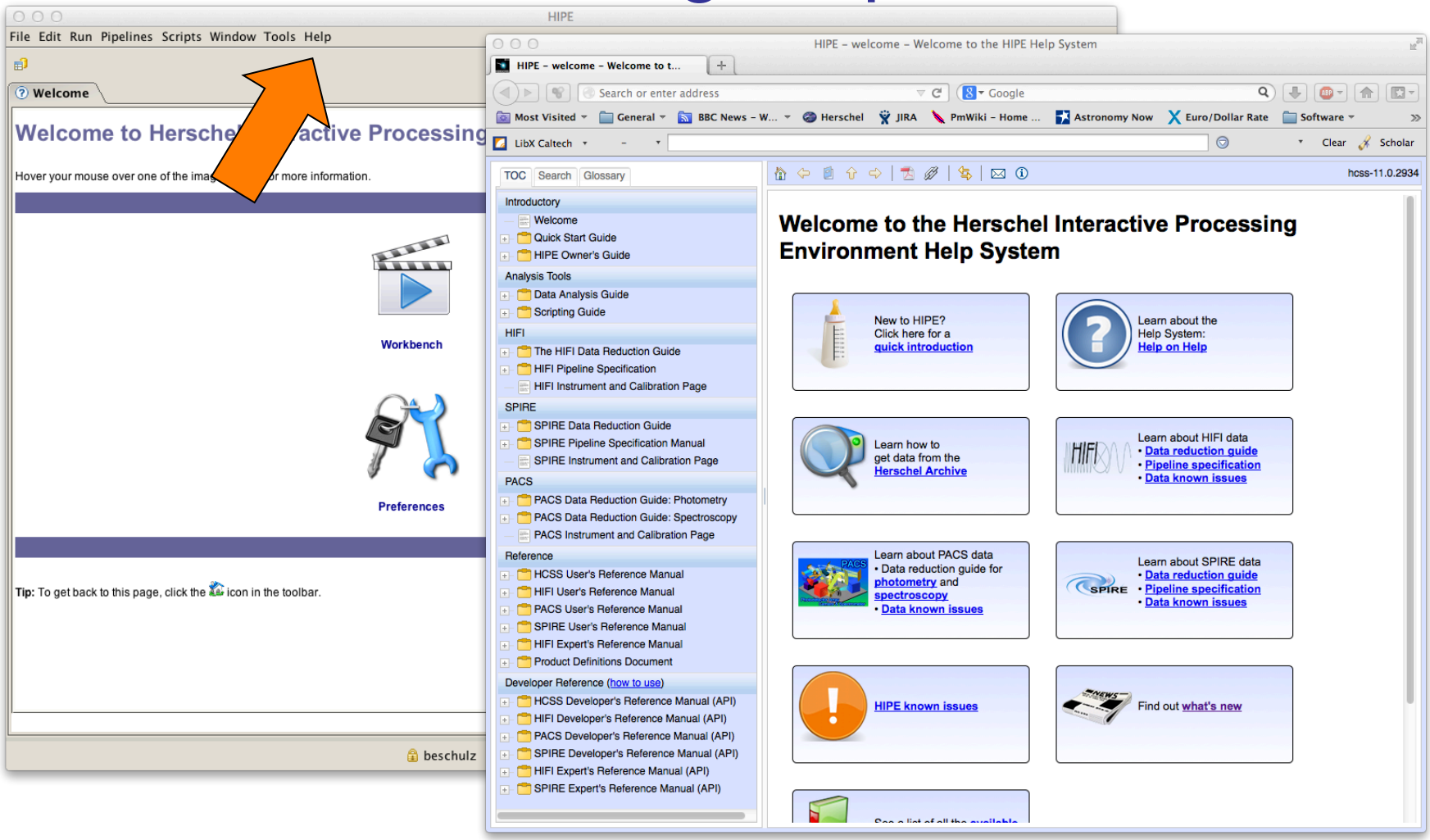
Busy indicator

Archive-connected indicator

Progress bar

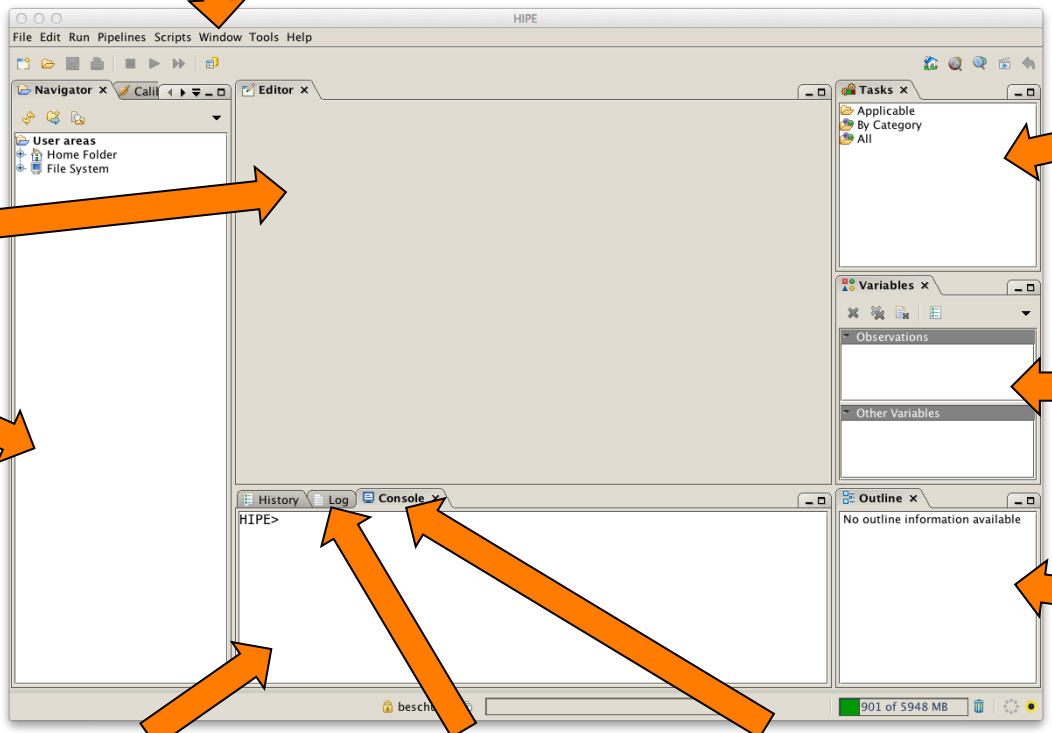
Memory bar

Getting Help



HIPE GUI Components (Views)

more here



Editor Window
Jython procedure editing
Data viewers

Navigator
Window to file system
Import data tarballs

Tasks View
Show available and applicable Tasks

Variables View
Shows variables of the session. Observations are shown in a special section.

Outline View
Shows internal structure of selected variable.

Console
Command line interface to Jython.

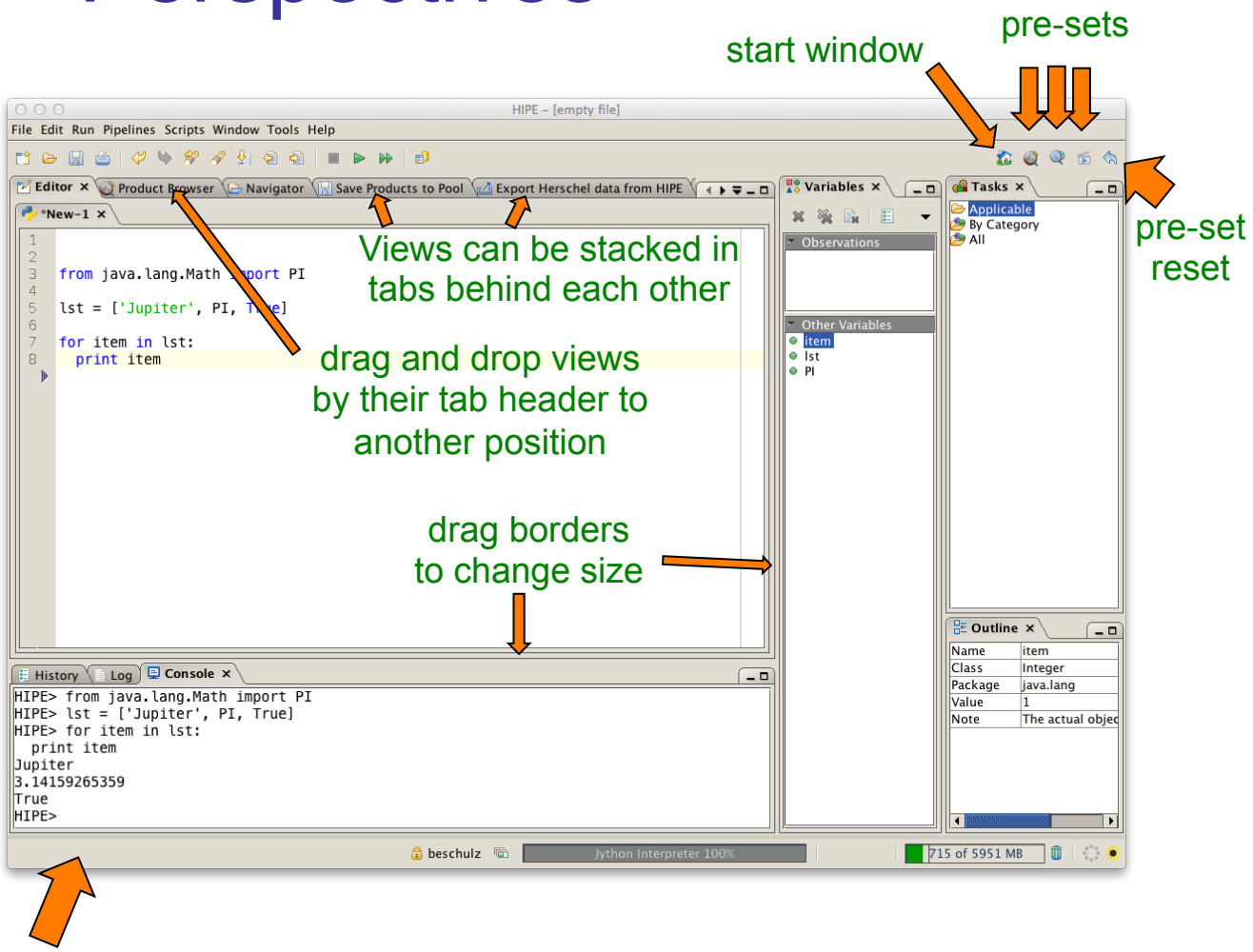
History View
Records command history and provides error information.

Log View
Logs output from tasks and procedures.

There are a few more views to discover…

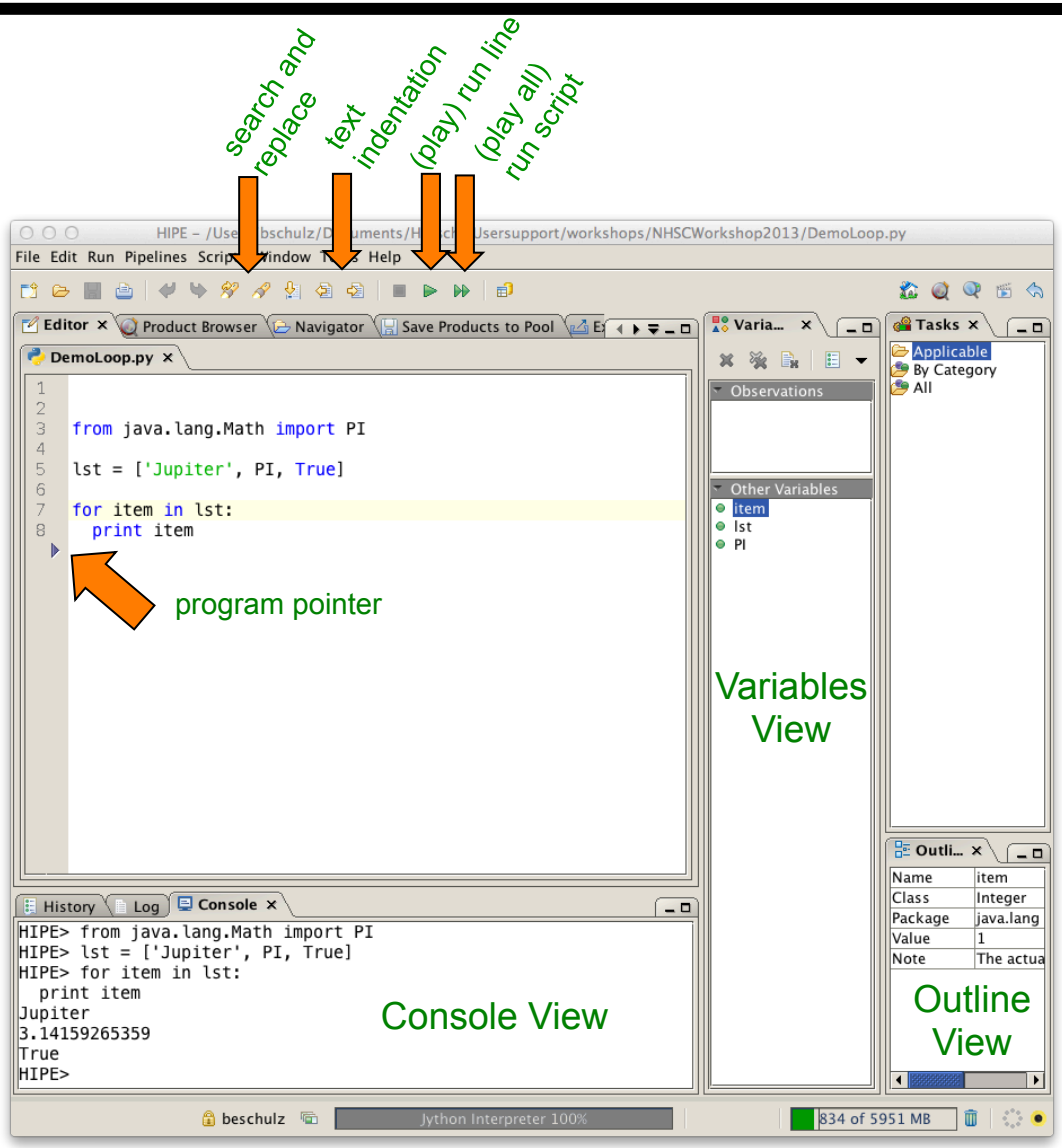
Perspectives

- Views can be arranged in many different ways, beside each other or stacked in tabs using drag-and-drop.
- Such an arrangement is called “Perspective”
- Three different pre-set perspectives are available by clicking the icons on the upper right of the panel.
- Any changes to the current perspective are remembered automatically in the last selected pre-set.
- The pre-sets can be reset to their original state.



Editor and Console Views

- The Console View works like a Python command line interface.
- Program code in the editor window can be executed using the green “play” arrows in the toolbar on the top.
- Lines selected by the small triangle (program pointer) to the left or highlighted code will be executed by clicking the single green arrow.
- The entire file will be executed clicking the green double arrow.
- More editor functions exist like search/replace, and shift text blocks right or left.
- Variables and Outline Views are updated when code is executed.



System Components

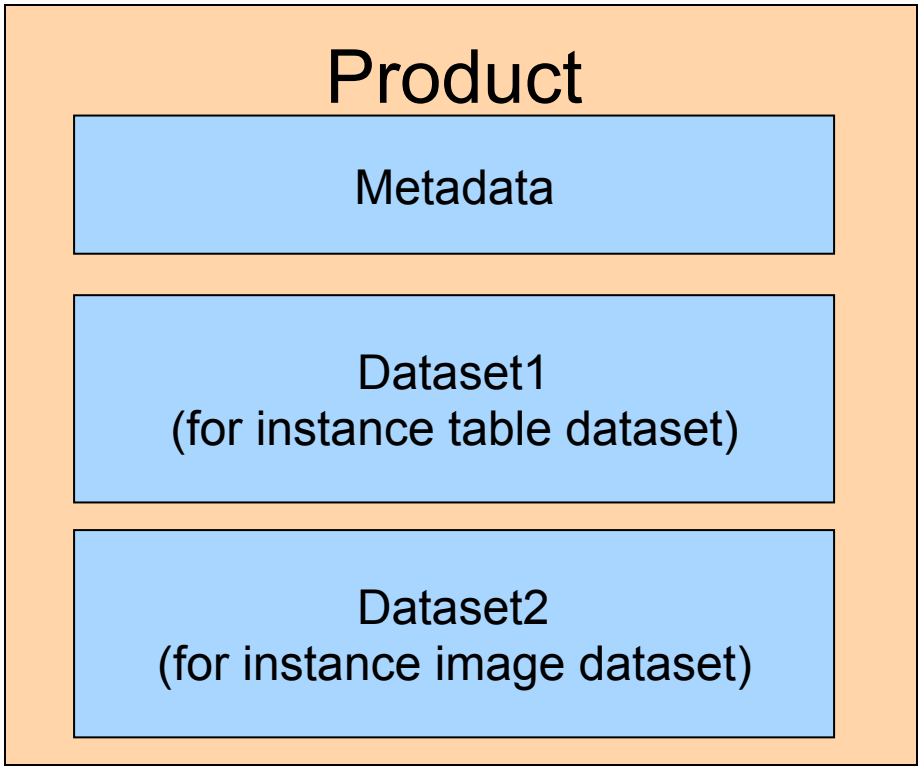
- HIPE GUI front end
- Jython interpreter
- Libraries (Numerics, Plot, IO, etc.)
- Pipeline and User scripts
- File system
- Databases (Pools, HSA)

Products

- In HIPE data is held within containers called products.
- These units were designed with the FITS file format in mind.
- A product can be exported from HIPE and saved as a FITS file.
- Datasets are saved by encapsulating them into a product.
- Every product has metadata.

General Product Structure

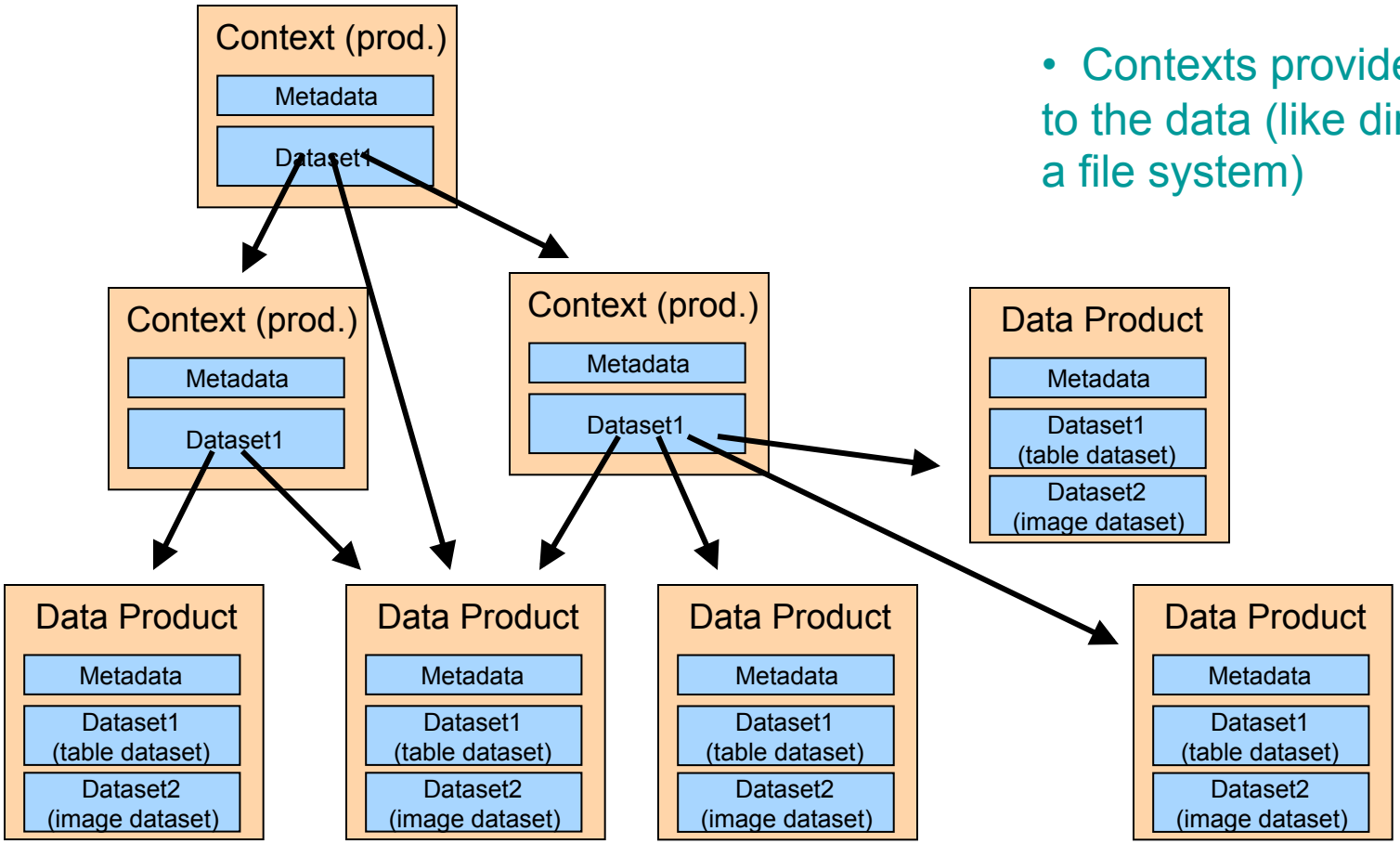
- Products are containers for datasets that can be stored within the HCSS system.
- A product can be stored in a pool or exported to a FITS file .



- Products contain:
  - Metadata,
  - Datasets
  - Processing history
- Types of datasets are:
  - Array dataset
  - Table dataset
  - Composite dataset
  - Spectrum1d
  - Spectrum2d
- Generic Product Types are:
  - SimpleImage
  - SimpleCube
  - SpectralSimpleCube
  - **Context**

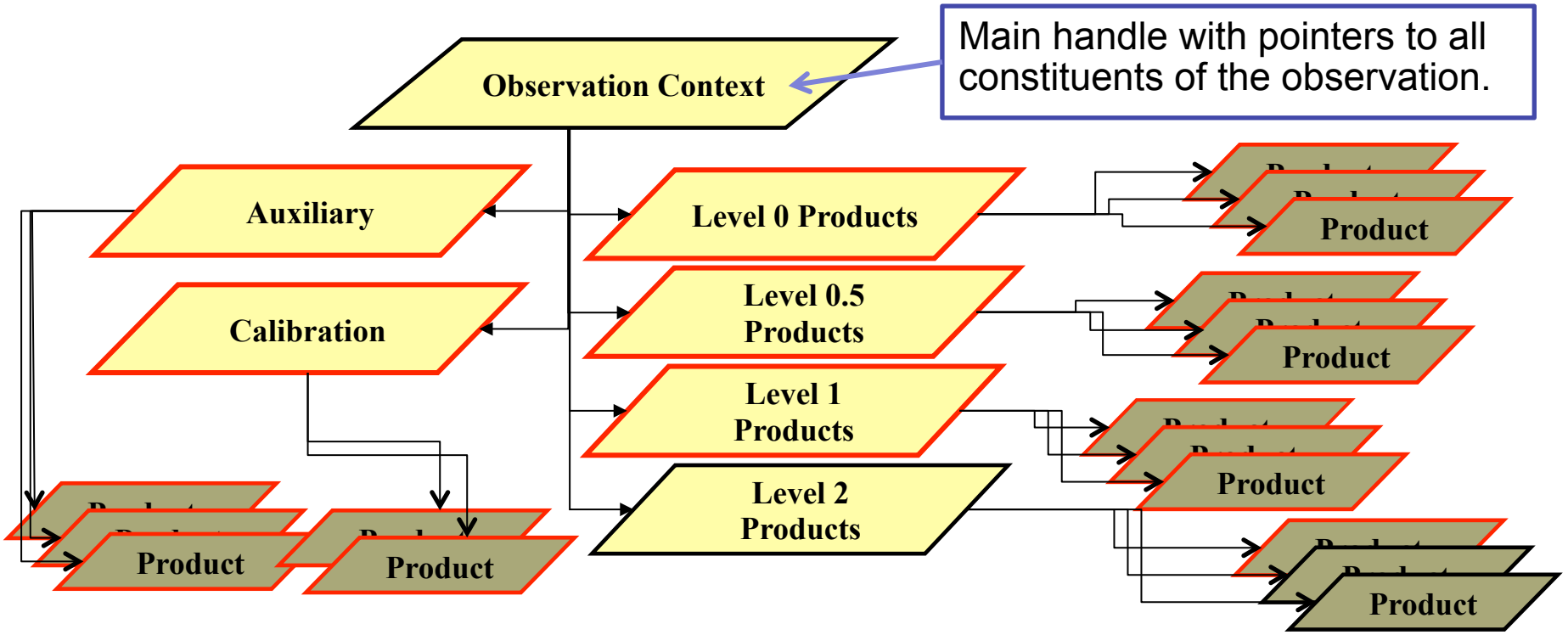
Contexts

- Contexts are products that point to other products.
- Contexts provide structure to the data (like directories in a file system)

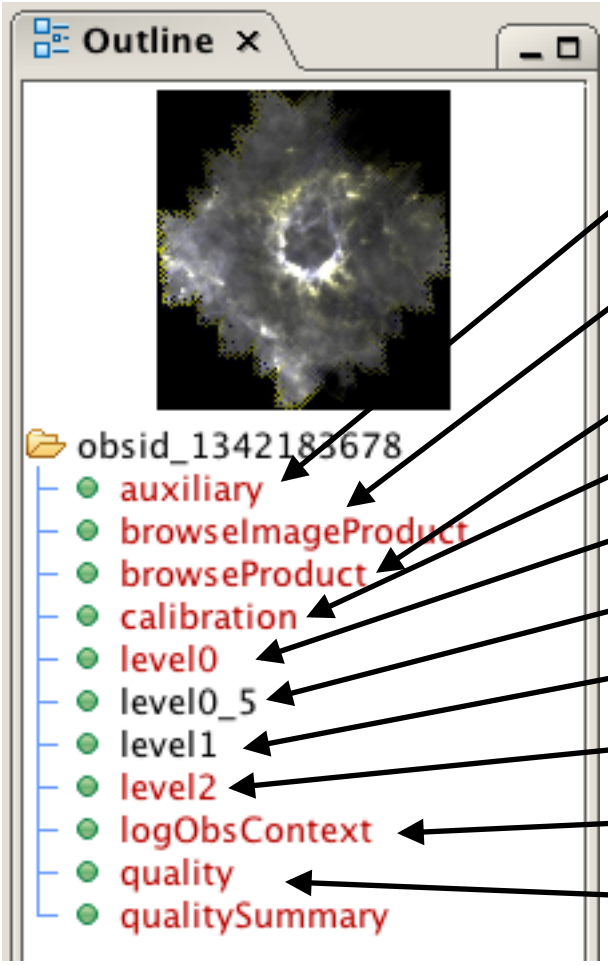


The Observation Context

- Observations are organized in product trees consisting of contexts and products with datasets.
- When loading an observation, initially only the observation context is loaded. Other products/contexts are loaded automatically if necessary ("lazy loading").
- Products not loaded yet into RAM are shown in red in the outline viewer of HIPE.
- When saving an observation context to a pool, all dependent products are saved as well.



Example: SPIRE Photometer Observation Context



These are all contexts!

Auxiliary data: Pointing, uplink info, etc…

Browse image product: Simple representation of final map for archive quicklook

Browse product: Another browse product

Calibration data: Products used for processing

Level 0: Raw unprocessed reformatted data

Level 0.5: Data converted to engineering units

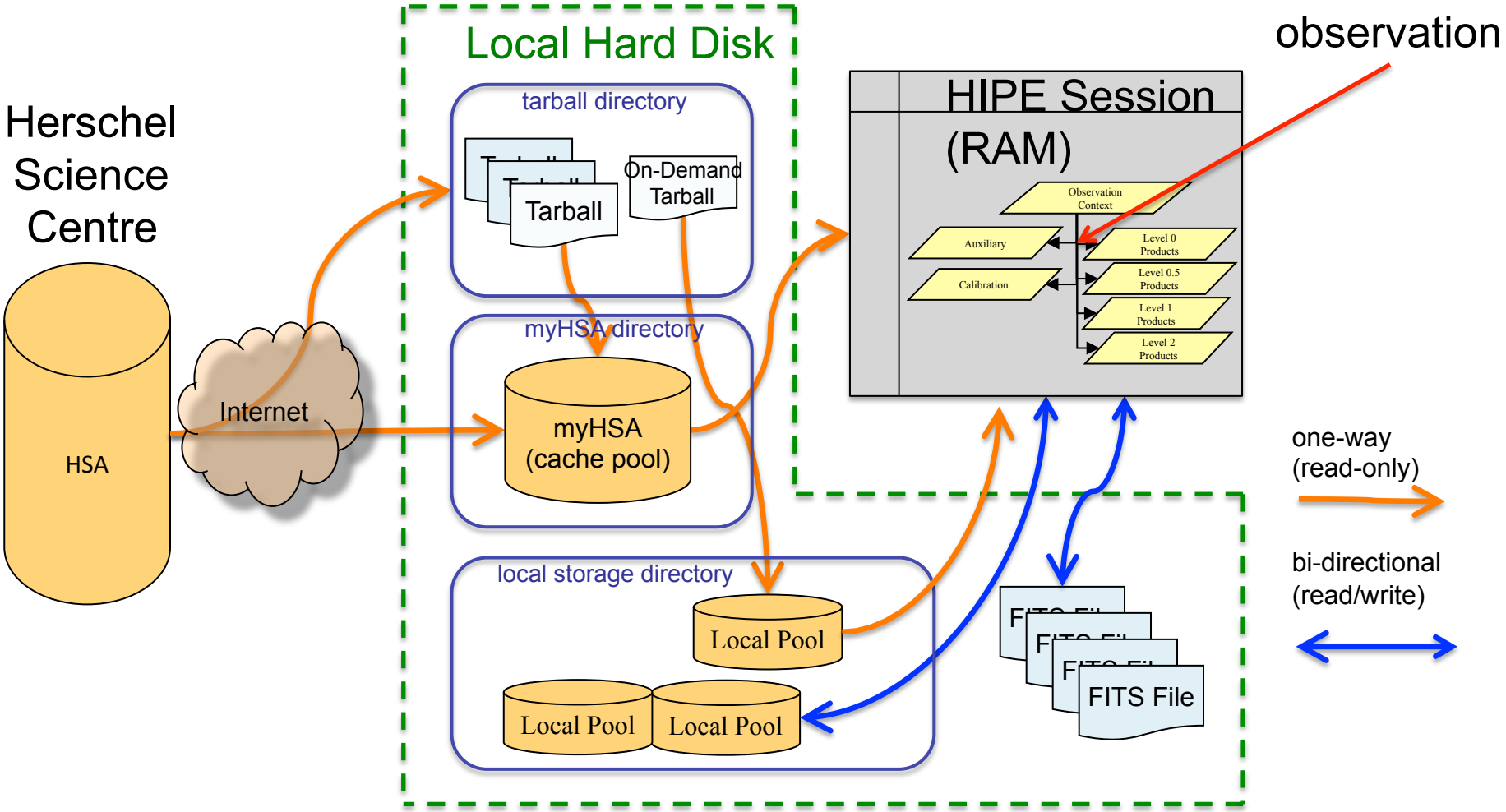
Level 1: Calibrated flux timelines [Jy/pixel]

Level 2: Images

Reduction Log: Data reduction history

Quality control data: Tells whether things went O.K.

HIPE Input/Output



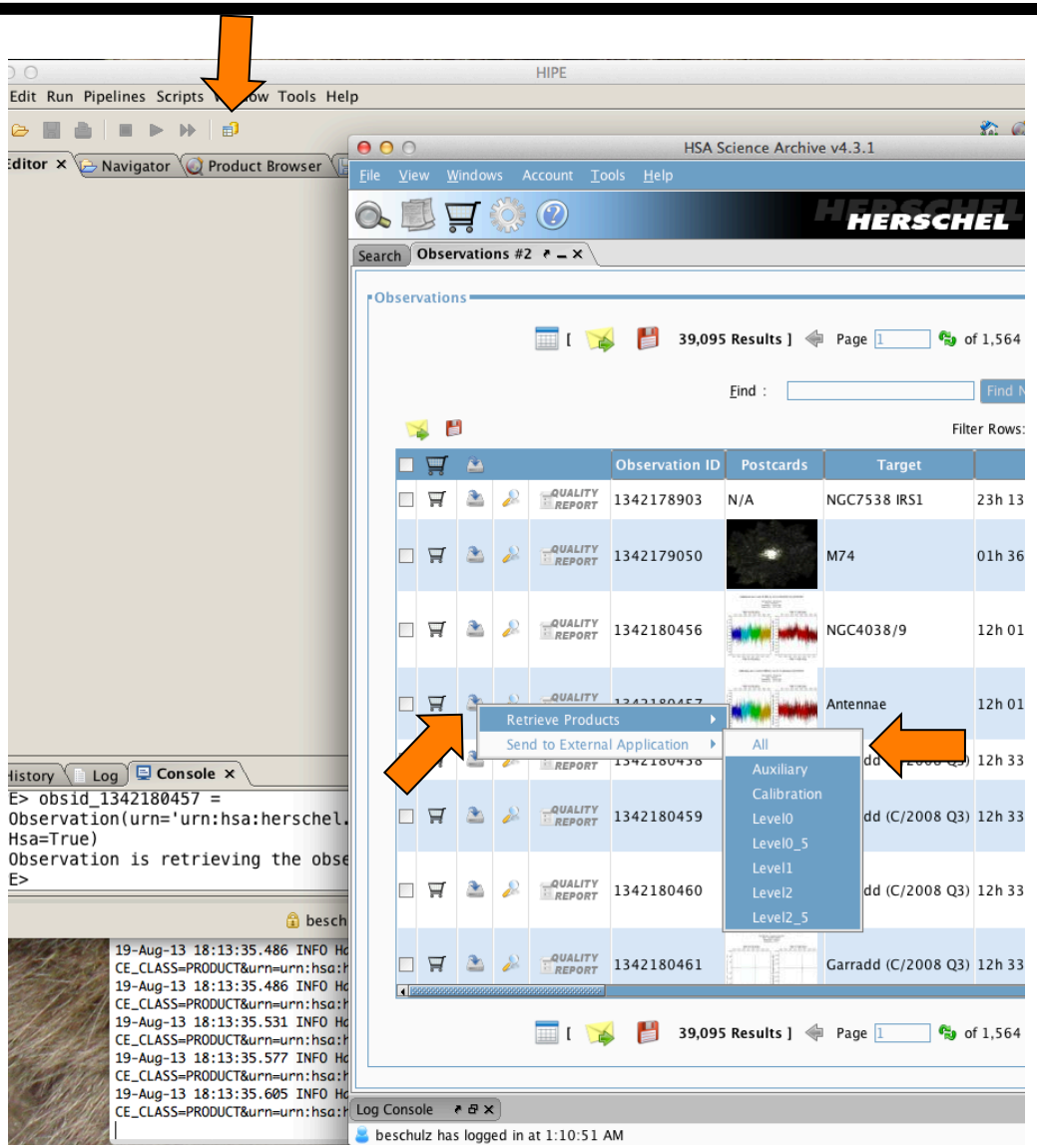
Data Input (Gimme my data)

- There are three important methods to load an observation into HIPE.
  - HSA User Interface (HUI)
    - Direct retrieval
    - Tarballs
  - getObservation()

Start HSA User Interface here

Data Input directly from HSA User Interface (HUI)

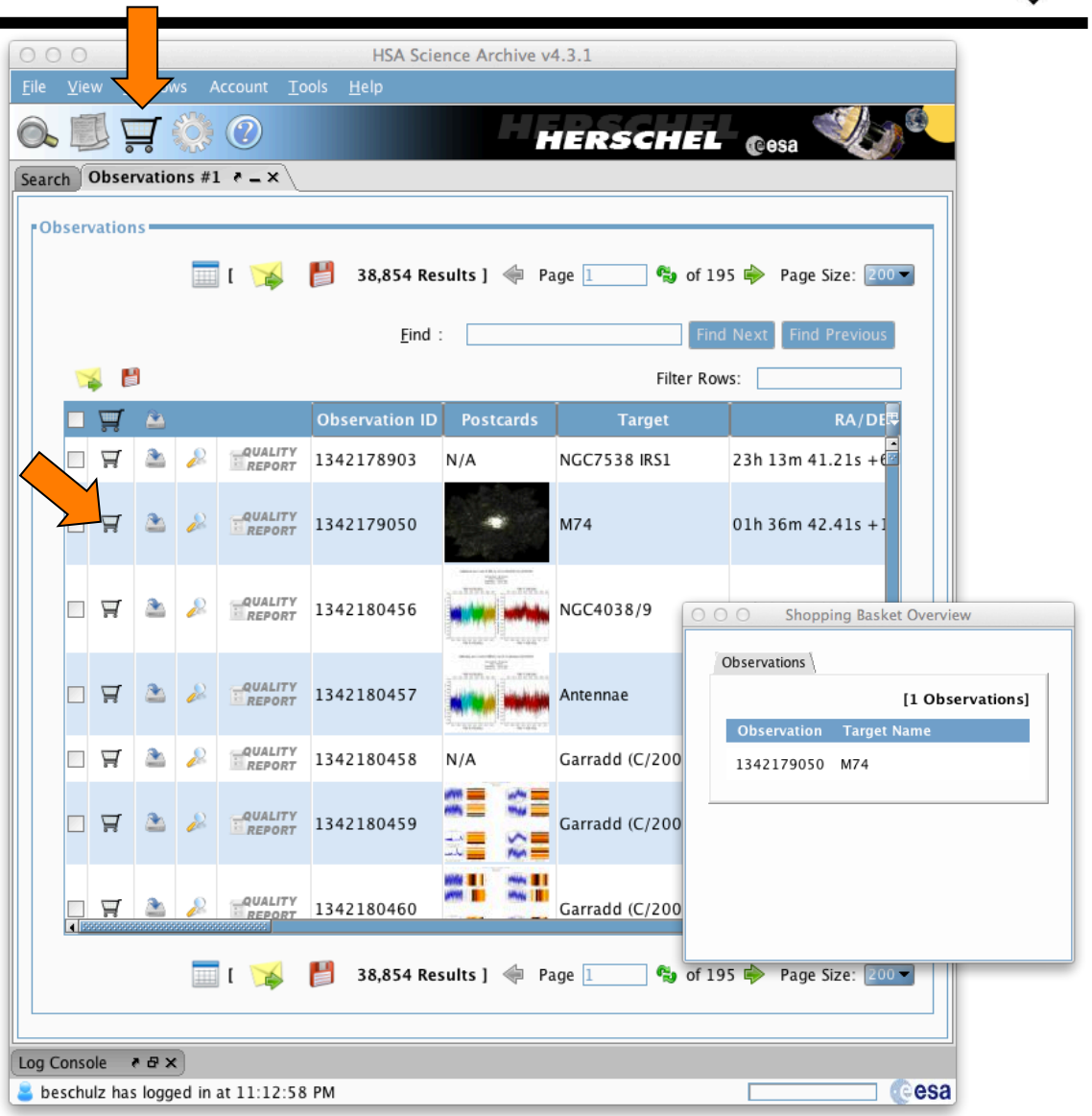
- Observations can be sent directly from the HSA User Interface to the HIPE session.
- Invoke it from the toolbar and select observations.
- Click on the hard disk symbol and select
  - Send to External Application
  - All
- The observation context will appear in the variables view of the HIPE session as a new variable.



Data Input via Tarball

- Observations can be packaged into tarballs, retrieved by FTP, and registered with the myHSA pool of HIPE.
- Click on the shopping basket symbol of the observation.
- Go to shopping basket, select observation, and submit request.
- Upon receipt of e-mail retrieve tarball with browser or FTP.
- Expand and place in directory where dataset can stay.
- Find xml file (Saturn Icon) of observation with Navigator and double-click to import.
- Observation appears as variable in the HIPE session and can also be found in the myHSA pool.

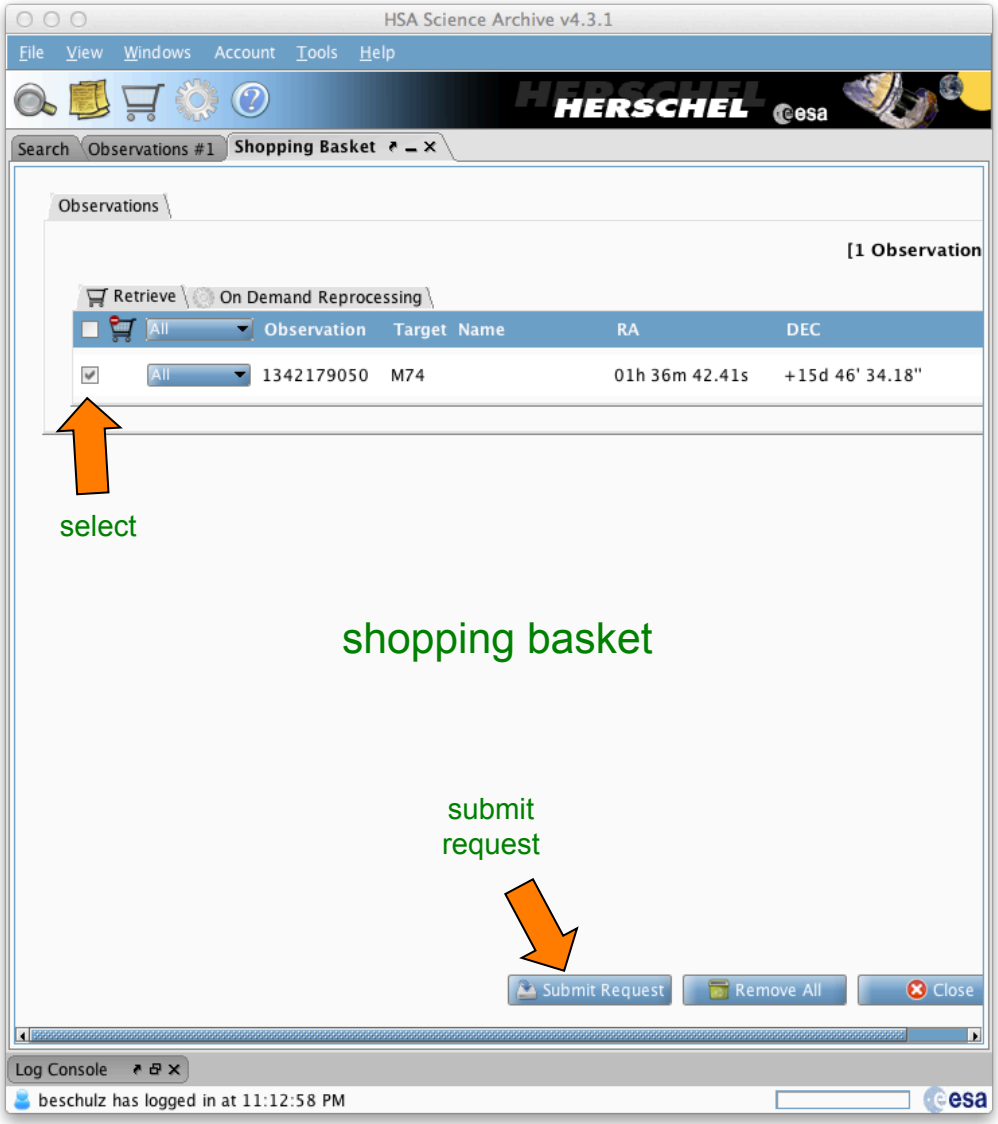
- Note: This import step needs only to be performed once. Next time the observation is immediately available using getObservation() or Product Browser.



Data Input via Tarball

- Observations can be packaged into tarballs, retrieved by FTP, and registered with the myHSA pool of HIPE.
- Click on the shopping basket symbol of the observation.
- Go to shopping basket, select observation, and submit request.
- Upon receipt of e-mail retrieve tarball with browser or FTP.
- Expand and place in directory where dataset can stay.
- Find xml file (Saturn Icon) of observation with Navigator and double-click to import.
- Observation appears as variable in the HIPE session and can also be found in the myHSA pool.

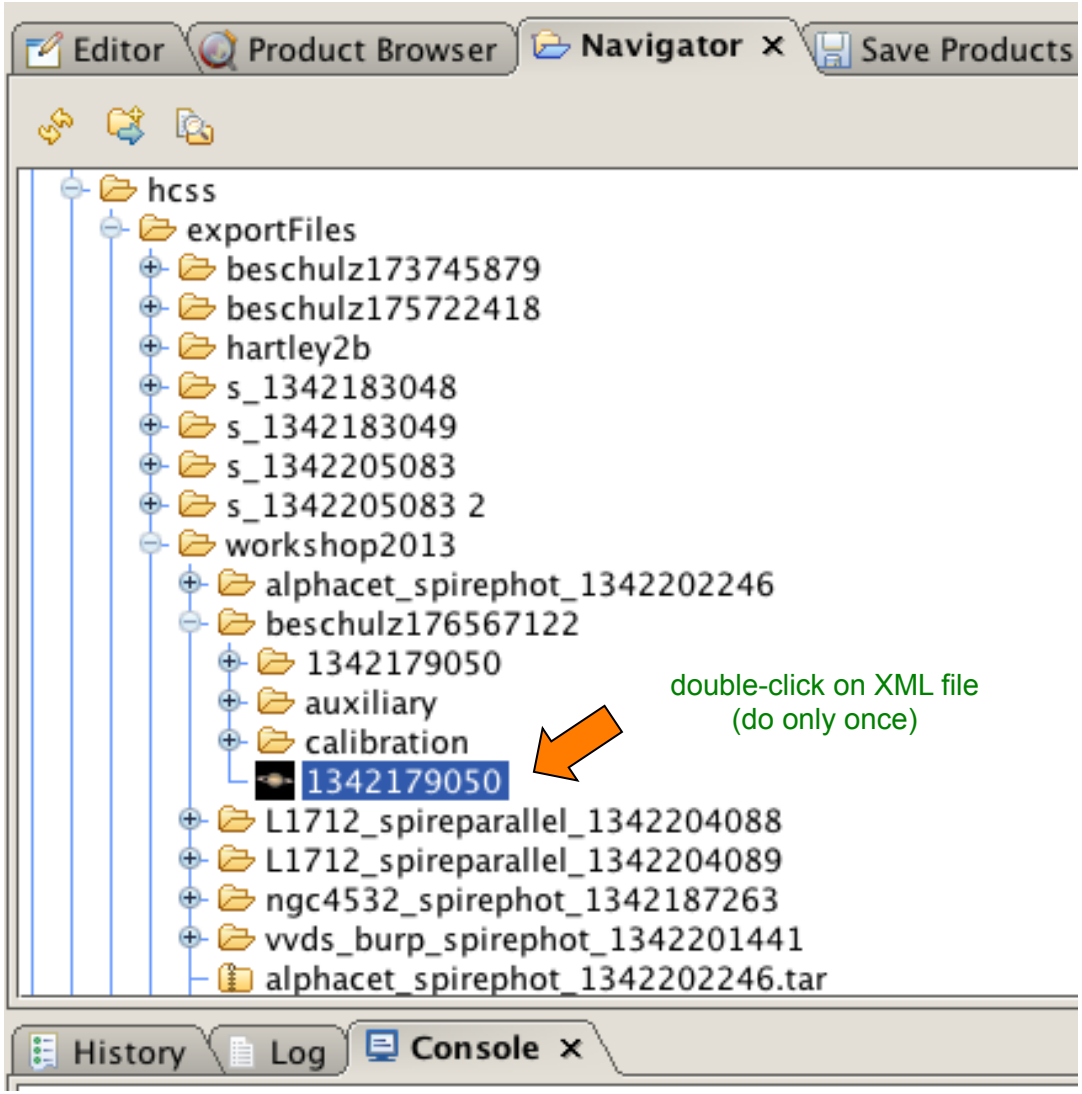
- Note: This import step needs only to be performed once. Next time the observation is immediately available using getObservation() or Product Browser.



Data Input via Tarball

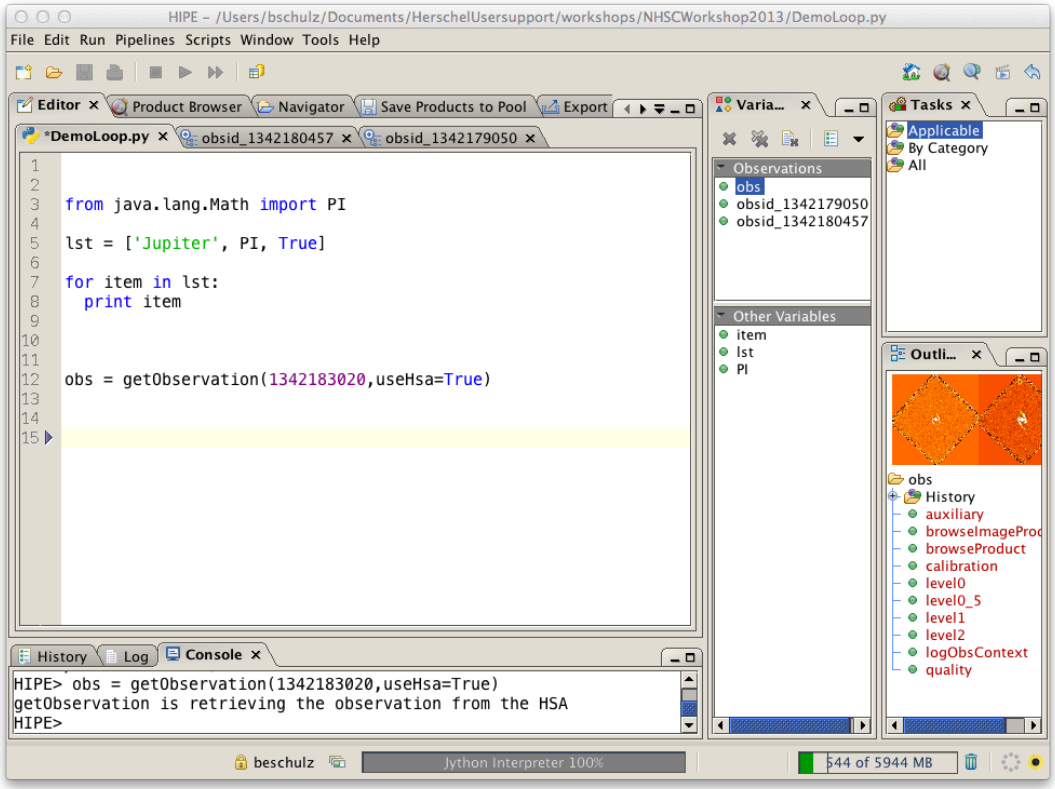
- Observations can be packaged into tarballs, retrieved by FTP, and registered with the myHSA pool of HIPE.
- Click on the shopping basket symbol of the observation.
- Go to shopping basket, select observation, and submit request.
- Upon receipt of e-mail retrieve tarball with browser or FTP.
- Place in directory where dataset can stay and expand it there.
- Find xml file (Saturn Icon) of observation with Navigator and double-click to import.
- Observation appears as variable in the HIPE session and can also be found in the myHSA pool.

- Note: This import step needs only to be performed once. Next time the observation is immediately available using getObservation() or Product Browser.



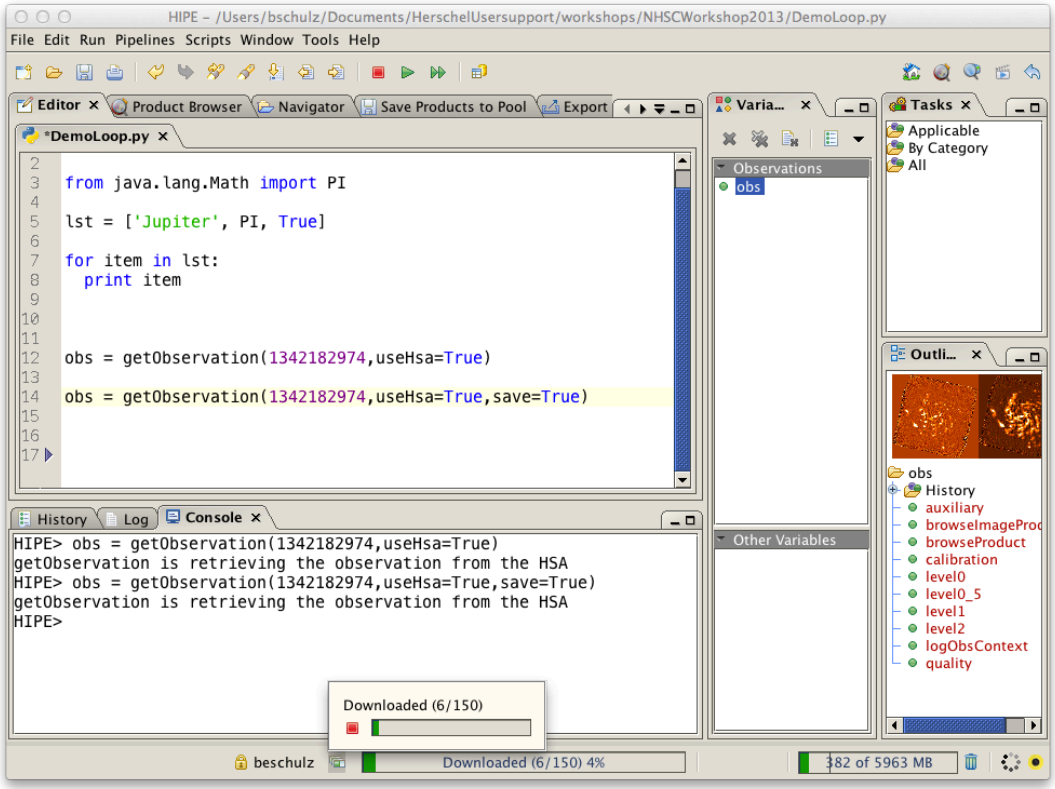
Data Input via getObservation()

- Observations can be retrieved directly from a script using the getObservation() task.
- With the option useHsa=True, the observation context is retrieved like with "send-to" from the HSA interface and placed into the HIPE session.
- With the additional option save=True, the entire observation is retrieved and saved in the myHsa pool. Next time it will already be on disk.
- This operation happens in the background so you can keep working.



Data Input via getObservation()

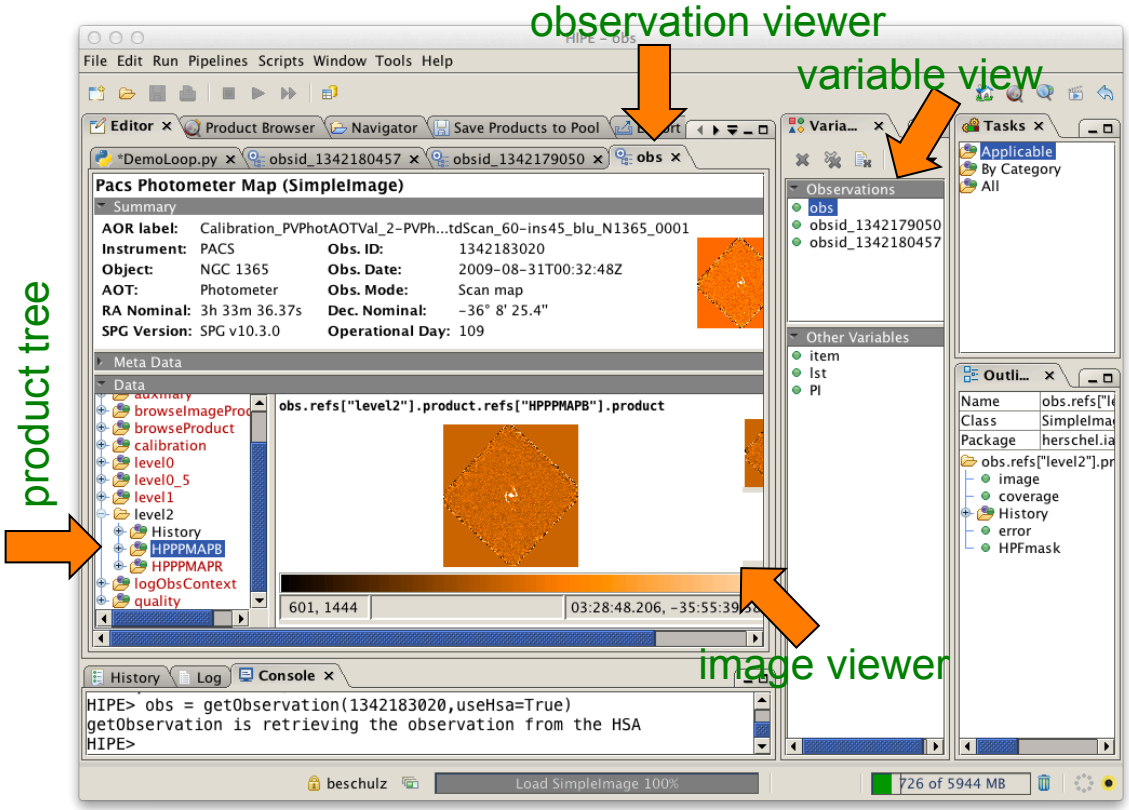
- Observations can be retrieved directly from a script using the getObservation() task.
- With the option useHsa=True, the observation context is retrieved like with "send-to" from the HSA interface and placed into the HIPE session.
- With the additional option save=True, the entire observation is retrieved and saved in the myHsa pool. Next time it will already be on disk.
- This operation happens in the background so you can keep working.



Inspecting Products

- Double-click on the observation context in the Variables View opens the Observation Viewer.
- Dependent contexts and products can be inspected by selection in the product tree to the left.
- Appropriate viewers for datasets are started automatically in the panel to the right of the product tree.
- Note that products only get actually transferred from the HSA when they are looked at (lazy-loading).

Observation Viewer (PACS example)



Data Viewing and Further Proceeds

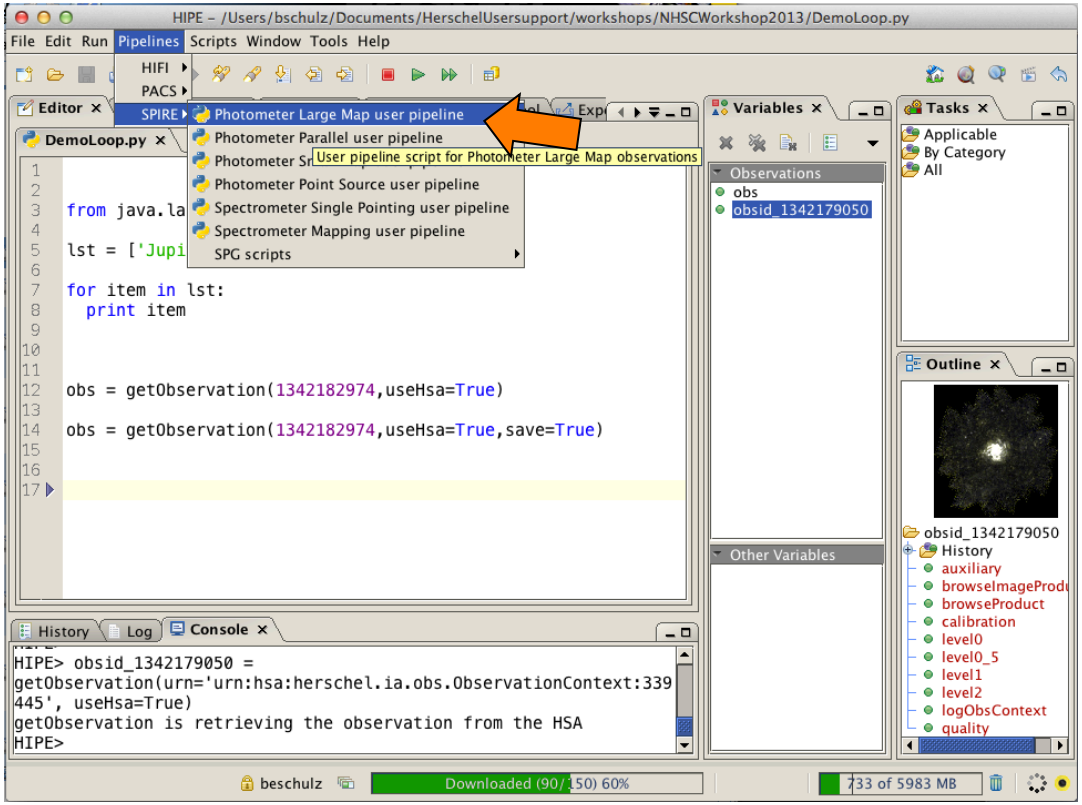
- More viewers are available and are often specific to a given instrument.
- These will be treated in the respective instrument sessions.
- Sometimes the data inspection reveals shortcomings in the data processing and the astronomer wishes to perform a data re-processing in HIPE.
- Data re-processing is usually done in Jython pipeline scripts or by specific “Tasks”.

Running a Pipeline Script

- Some custom scripts are available in the HIPE menu already.
- Locations:
  - Under "Pipelines" the main pipeline scripts can be found that are used to produce the archive results.
  - Under "Scripts" more Jython scripts are available for more specialized non-standard purposes.
- Script Execution in the Editor View:
  - The single green arrow executes one statement or block at the lowest level.
  - The green double arrow executes the entire file.
  - Indented blocks like for-loops or if-then statements can not be stepped through line by line.
  - The pause() statement sets break-points and helps with debugging inside of indented blocks.
  - Single lines or groups of lines inside blocks can be executed with limitations by marking them blue and hitting the single green arrow.

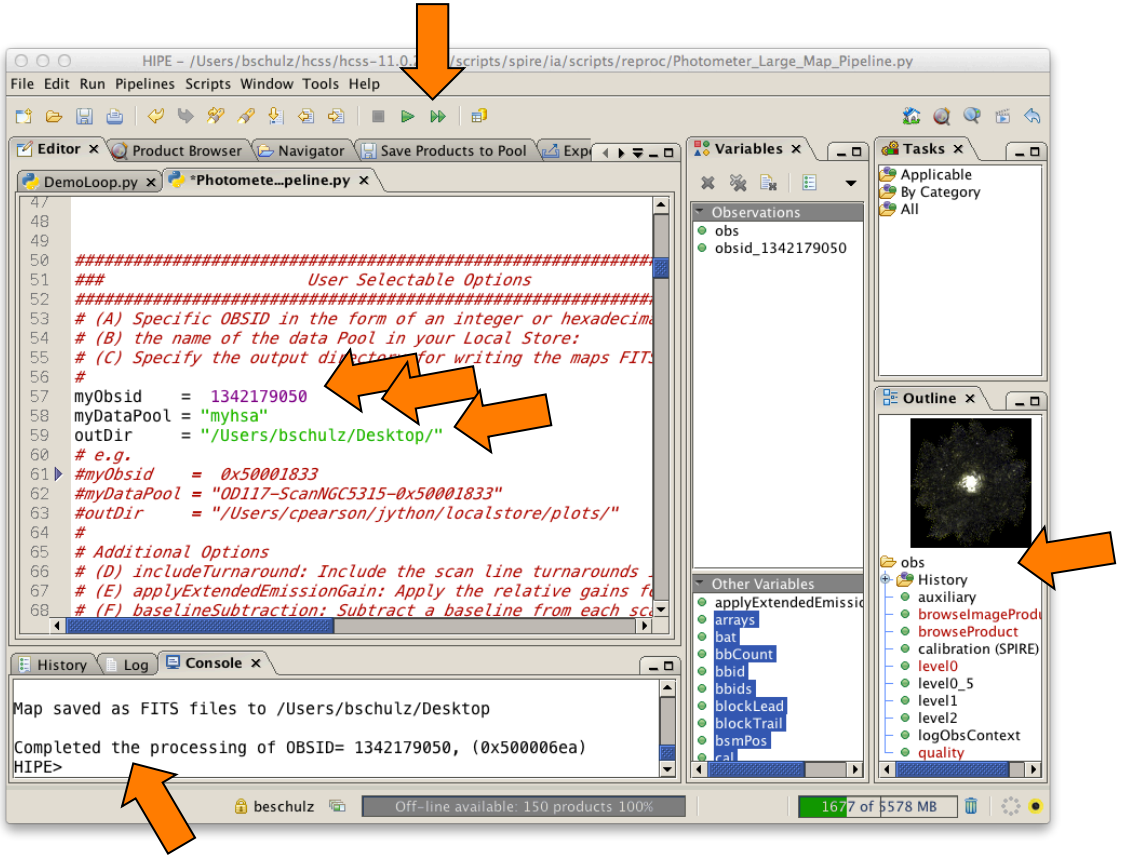
Example: Running the SPIRE Pipeline

- Select in the Pipelines Menu: “Photometer Large Map user pipeline”
- Edit the script without saving and insert
  - myObsid=1342182974
  - myDataPool=‘myhsa’
  - outDir=‘<output directory>’
- Run the script by hitting the green double arrow in the top toobar.
- After a while the updated observation will appear as obs under Variables and in the Outline view.



Example: Running the SPIRE Pipeline

- Select in the Pipelines Menu: "Photometer Large Map user pipeline"
- Edit the script without saving and insert
  - myObsid=1342182974
  - myDataPool='myhsa'
  - outDir='<output directory>'
- Run the script by hitting the green double arrow in the top toobar.
- After a while the updated observation will appear as obs under Variables and in the Outline view.

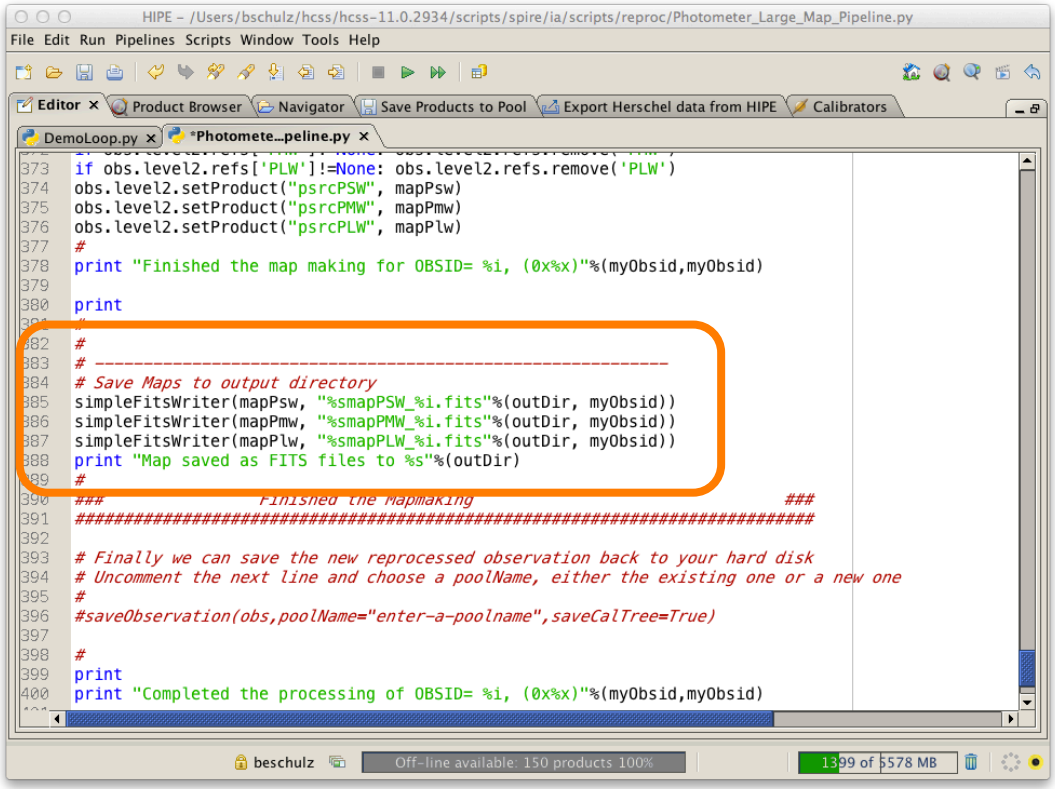


Saving Products in FITS Files

- If only a few products are of interest after processing is complete, each product can be saved to a FITS file.
  - Right-click on the product (e.g. a map) and select Send-to → FITS-file.
  - In the upcoming GUI of the task simpleFitsWriter() enter a filename with path and hit the “Accept” button.
  - The equivalent command appears in the Command View and can be re-used in a Jython script.

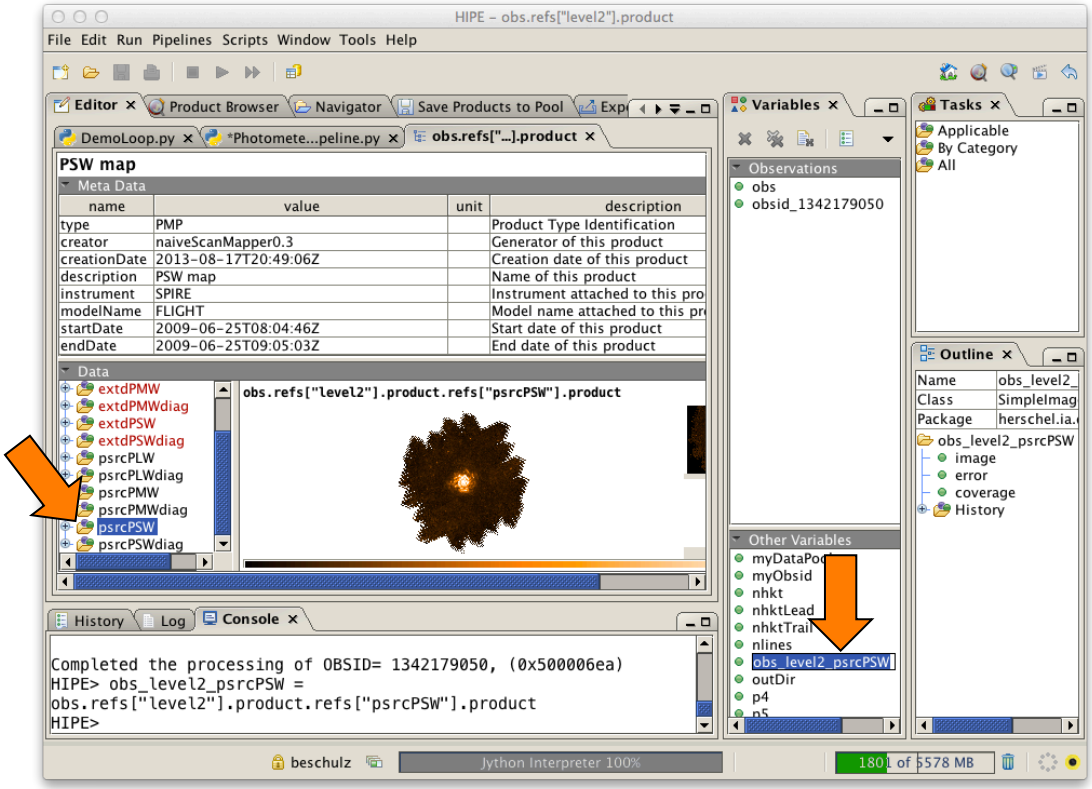
Saving Products to FITS files

- The script already saved the three map products as FITS files.
- To try using the GUI:
- Double-click Level 2 in the Outline view.
- Drag a map from Level 2 into the variable space.
- Right click on the new variable and select “Send-to” and “FITS-file”.
- In the upcoming GUI enter a filename with path and hit the “Accept” button.
- Note that you just used a Task from its default GUI (more about this later).
- Note also that the scripted version of what you just did appears in the Console View.



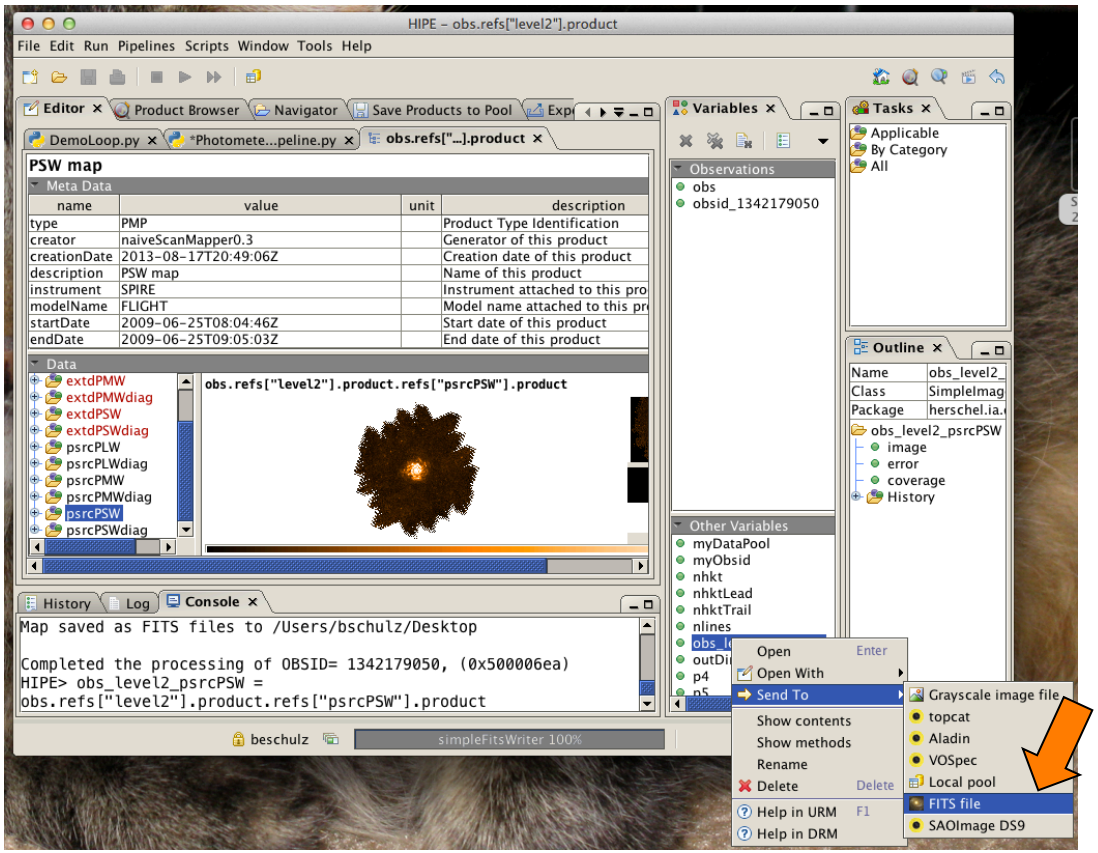
Saving Products to FITS files

- The script already saved the three map products as FITS files.
- To try using the GUI:
- Double-click Level 2 in the Outline view.
- Drag a map from Level 2 into the variable space.
- Right click on the new variable and select “Send-to” and “FITS-file”.
- In the upcoming GUI enter a filename with path and hit the “Accept” button.
- Note that you just used a Task from its default GUI (more about this later).
- Note also that the scripted version of what you just did appears in the Console View.



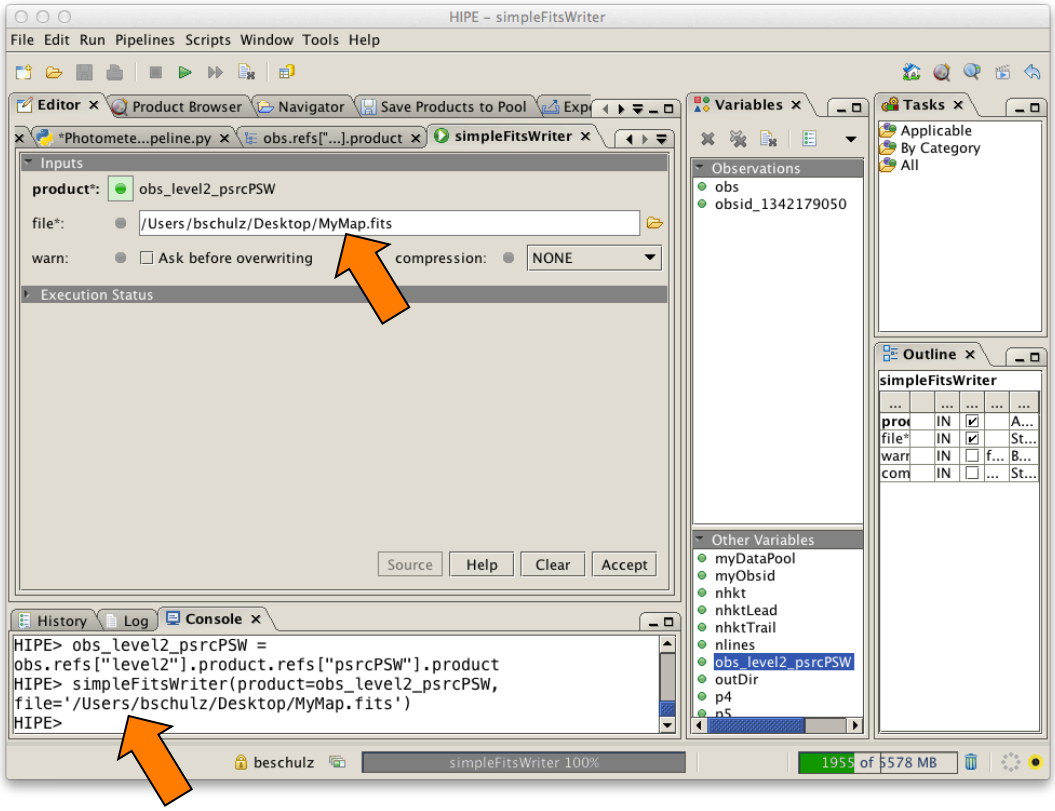
Saving Products to FITS files

- The script already saved the three map products as FITS files.
- To try using the GUI:
- Double-click Level 2 in the Outline view.
- Drag a map from Level 2 into the variable space.
- Right click on the new variable and select “Send-to” and “FITS-file”.
- In the upcoming GUI enter a filename with path and hit the “Accept” button.
- Note that you just used a Task from its default GUI (more about this later).
- Note also that the scripted version of what you just did appears in the Console View.



Saving Products to FITS files

- The script already saved the three map products as FITS files.
- To try using the GUI:
- Double-click Level 2 in the Outline view.
- Drag a map from Level 2 into the variable space.
- Right click on the new variable and select “Send-to” and “FITS-file”.
- In the upcoming GUI enter a filename with path and hit the “Accept” button.
- Note that you just used a Task from its default GUI (more about this later).
- Note also that the scripted version of what you just did appears in the Console View.



Saving Observations in Local Pools

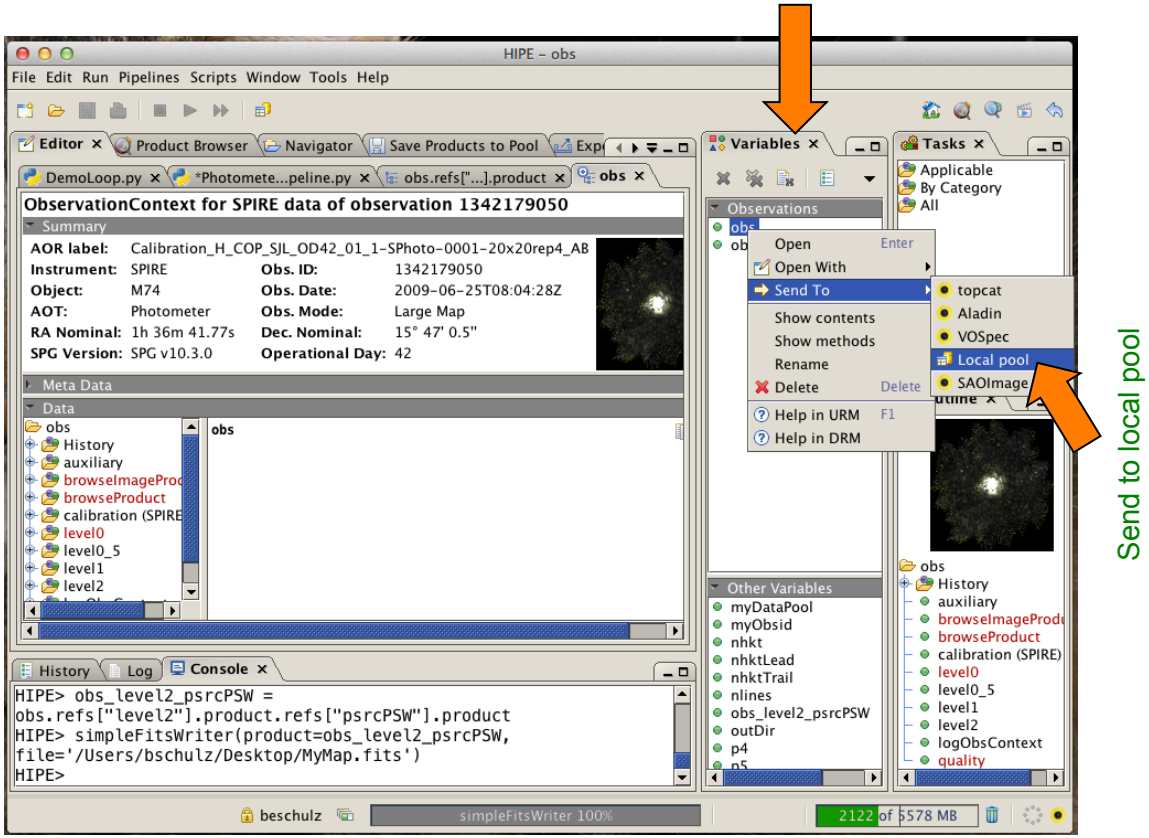
- If you work on an observation and update many of its products, and you want to save a copy of your work for the next session, saving single FITS files is impractical.
- In this case the better way is to use a Local Pool.
- Local Pools are databases for products and contexts that keep the product tree structure intact.
- Saving and retrieving entire observation trees can be performed very easily using the observation context as a handle.

Saving Observations in Pools

- Select the observation context in the Variable View.
- Select “Send To” and “Local pool”.
- Find observation in list.
- Add some text (Tag) for identification.
- Enter a destination pool name (not existing ones will be created new).
- Hit ‘Save’ button.
- Note the command line representation:

```
saveProduct(product=obs,
pool='myTestPool',
tag='obs 1342179050 my
last reduction')
```

Variables View

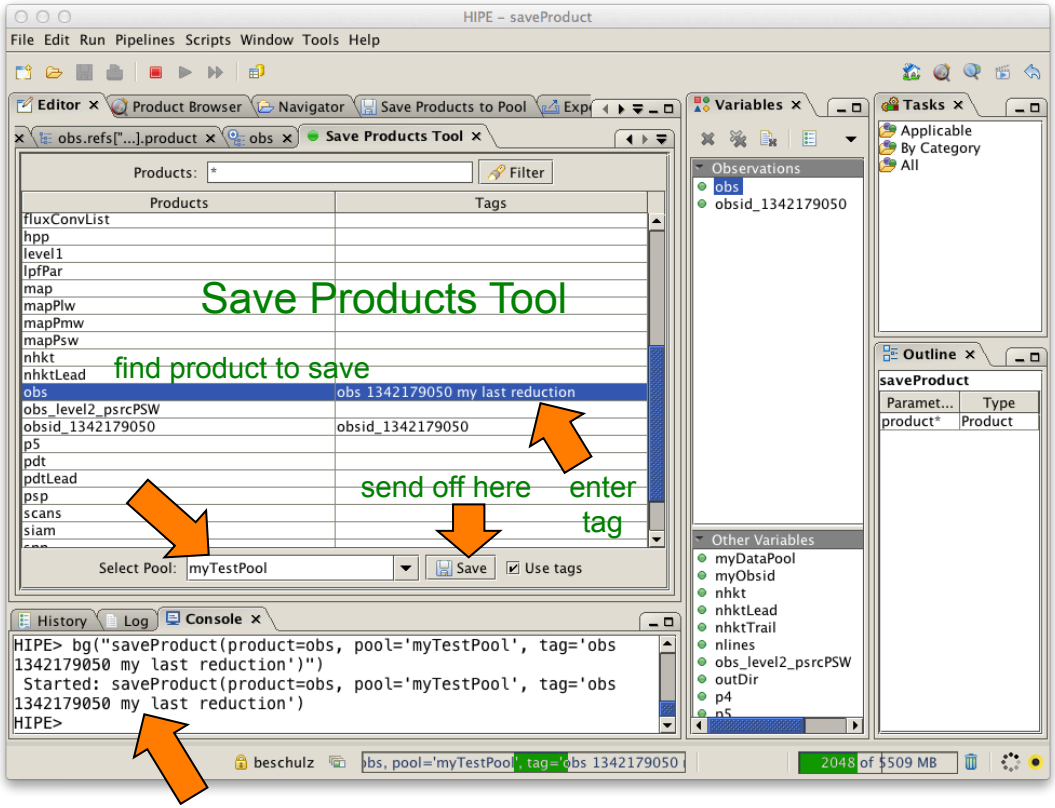


Send to local pool

Saving Observations in Pools

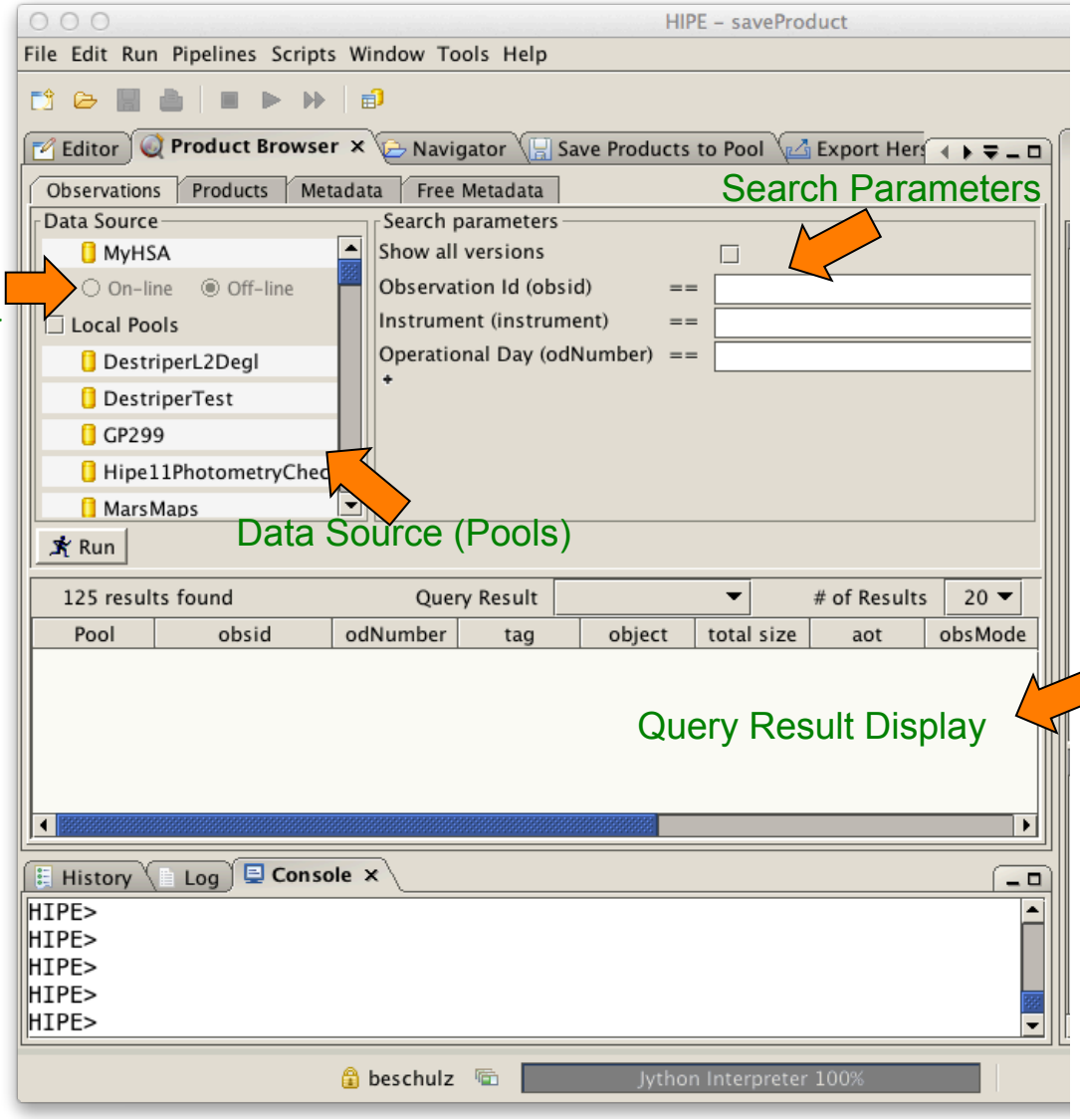
- Select the observation context in the Variable View.
- Select “Send To” and “Local pool”.
- Find observation in list.
- Add some text (Tag) for identification.
- Enter a destination pool name (not existing ones will be created new).
- Hit ‘Save’ button.
- Note the command line representation:

```
saveProduct(product=obs,
pool='myTestPool',
tag='obs 1342179050 my
last reduction')
```



The Product Browser

- The Product Browser can be found in “Window” as a view under “Data Access”.
- The main panels are “Data Source”, “Search parameters”, and “Query Result Display”
- Data Source shows all the pools in your storage directory and the myHsa pool.
- On the right some search criteria can be chosen.
- The “On-line” button enables querying the HSA through the internet if an observation isn’t present locally in the MyHSA pool.

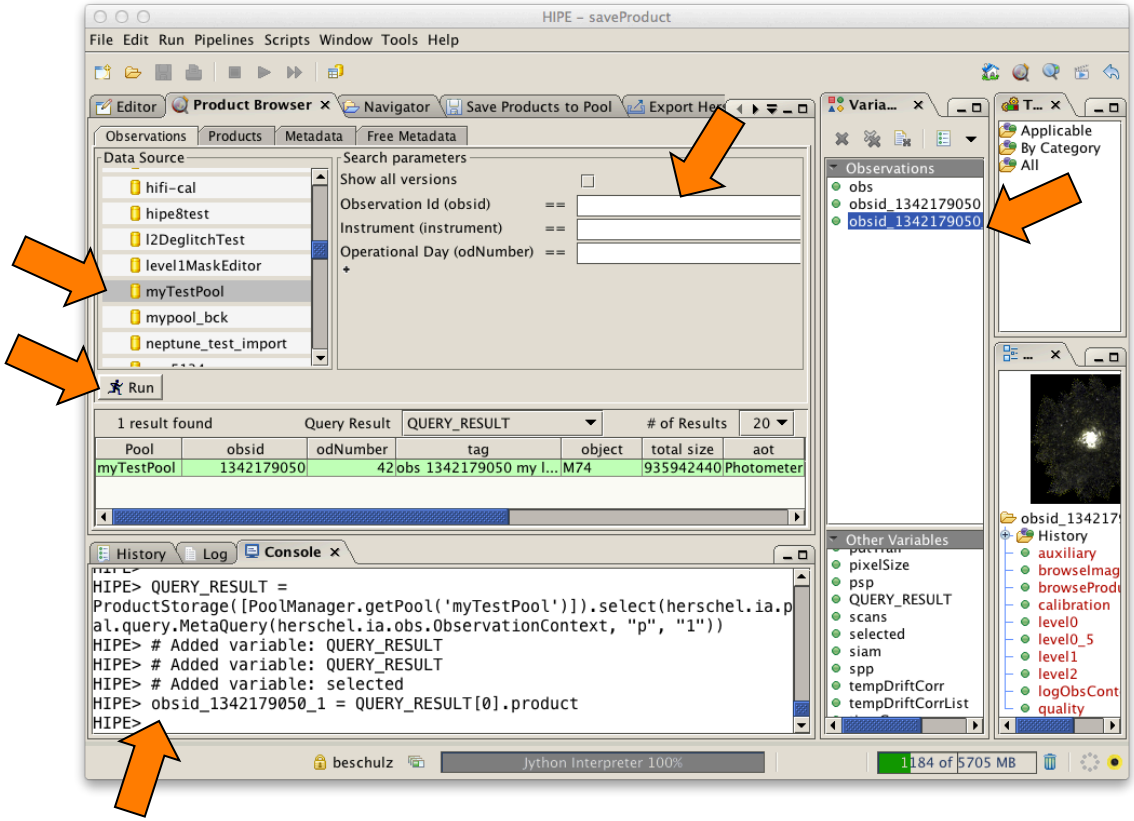


Managing Observations and Pools

- Pools can be created, inspected, renamed, moved, exported, and deleted.
- The Product Browser helps to easily visualize and manage the content of pools.
- The myHSA pool is special:
  - MyHsa acts as a local cache between the HSA at the Herschel Science Center and your HIPE session on your computer.
  - All pools can be written to, except for the "myHsa" pool.
  - The other pools are intended for saving results that are different from those in the HSA, i.e. products from your own processing and analysis.
  - The data transfer between the HSA and the myHSA pool is optimized.
  - Saving observations that were retrieved straight from the HSA into a pool, is less efficient.
- The Product Browser can be used to retrieve observations from the archive by querying myHsa with the On-line option on.

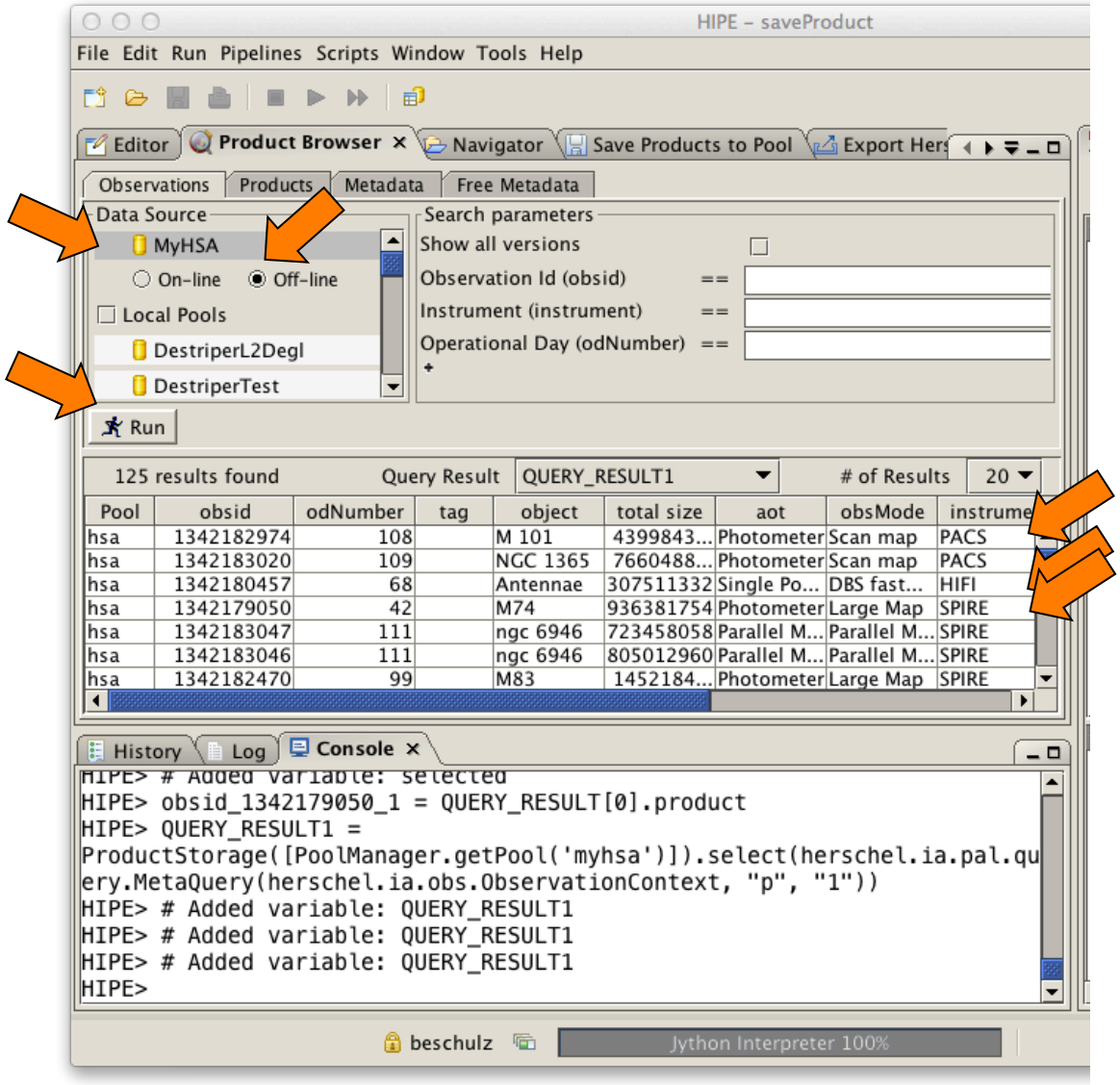
Pool Content and Retrieval

- Find and select the newly created pool “myTestPool”.
- Hit the “Run” button.
- The query result list will show one line representing the observation that was just saved.
- Double-clicking the observation will retrieve the observation context.
- Note the more complex script equivalent in the Console View.



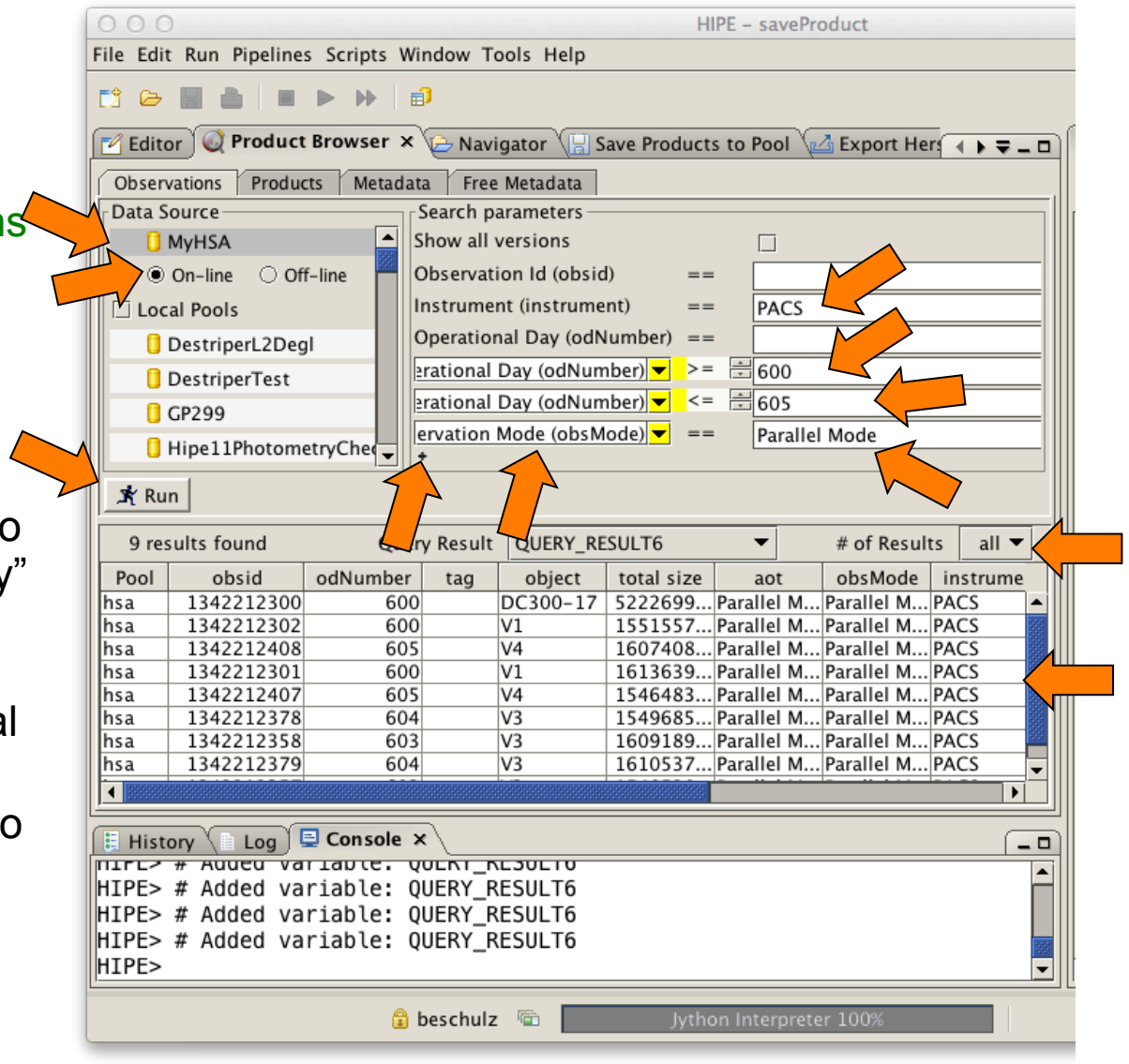
MyHSA Access

- To access observations in the local myHSA pool, select MyHSA
- Ensure "Off-line" is selected.
- Hit the "Run" button.
- Note the presence of the earlier imported observations from SPIRE, HIFI and PACS.



On-line Archive Access Example

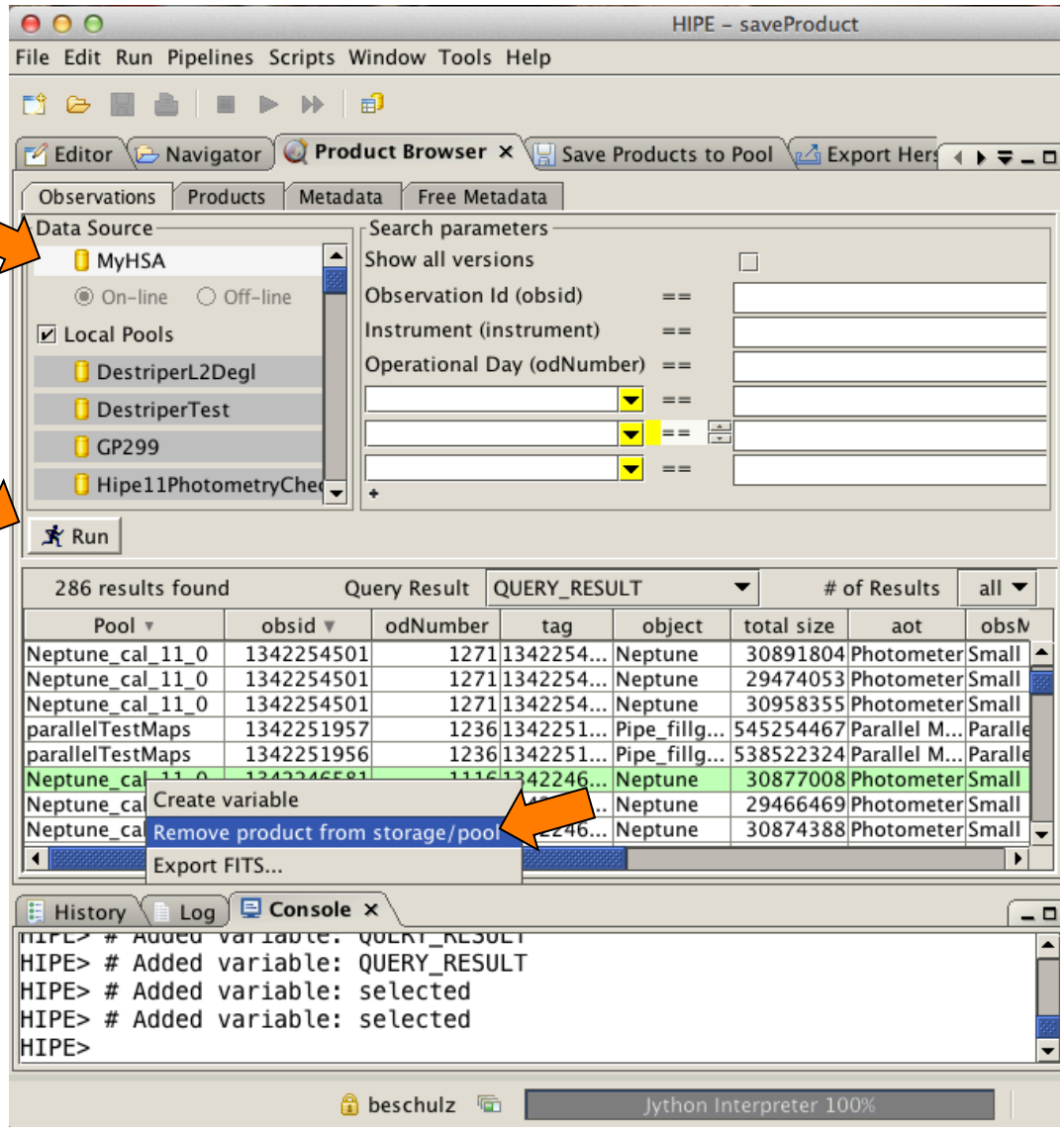
- Let's find all parallel mode PACS observations on observational days 600 and 605.
- Switch to "On-line"
- Enter "PACS" into Instrument.
- Use the + sign to add two more "Observational Day" search criteria and one "Observation Mode".
- Select appropriate logical operators and numbers.
- Enter "Parallel Mode" into Observation Mode
- Hit "Run".
- Select "all" to show the full list of results.



Managing Observations

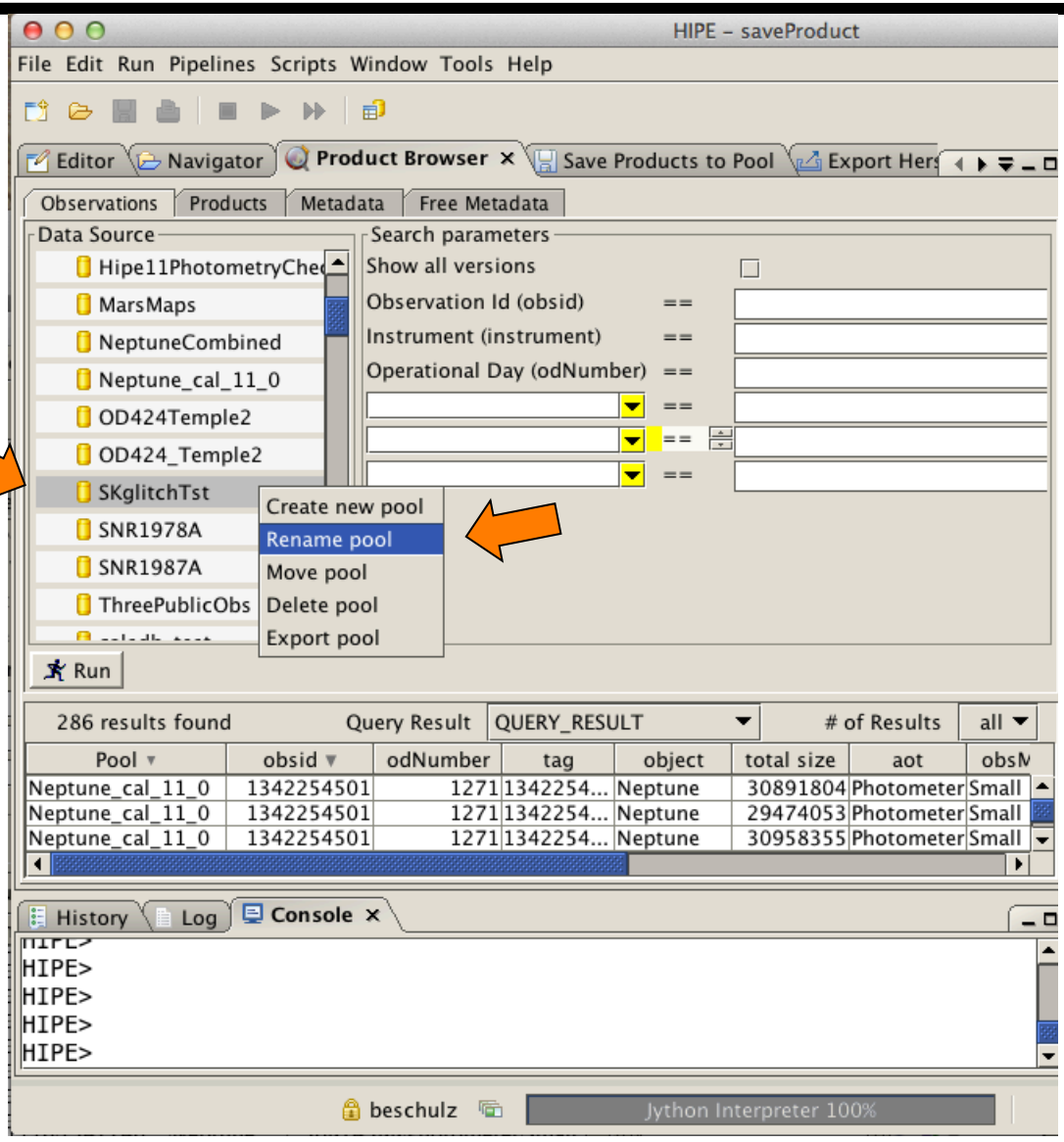
- Find all observations in all local pools.
- De-select “MyHSA” by clicking on another pool.
- Select “Local Pools”.
- Hit “Run”.
- Right click on an observation.
- The menu allows to:

1. load an observation context into RAM (create variable),
2. Delete an observation from a pool (Remove product….),
3. Export FITS (not useful because it is just a context.).



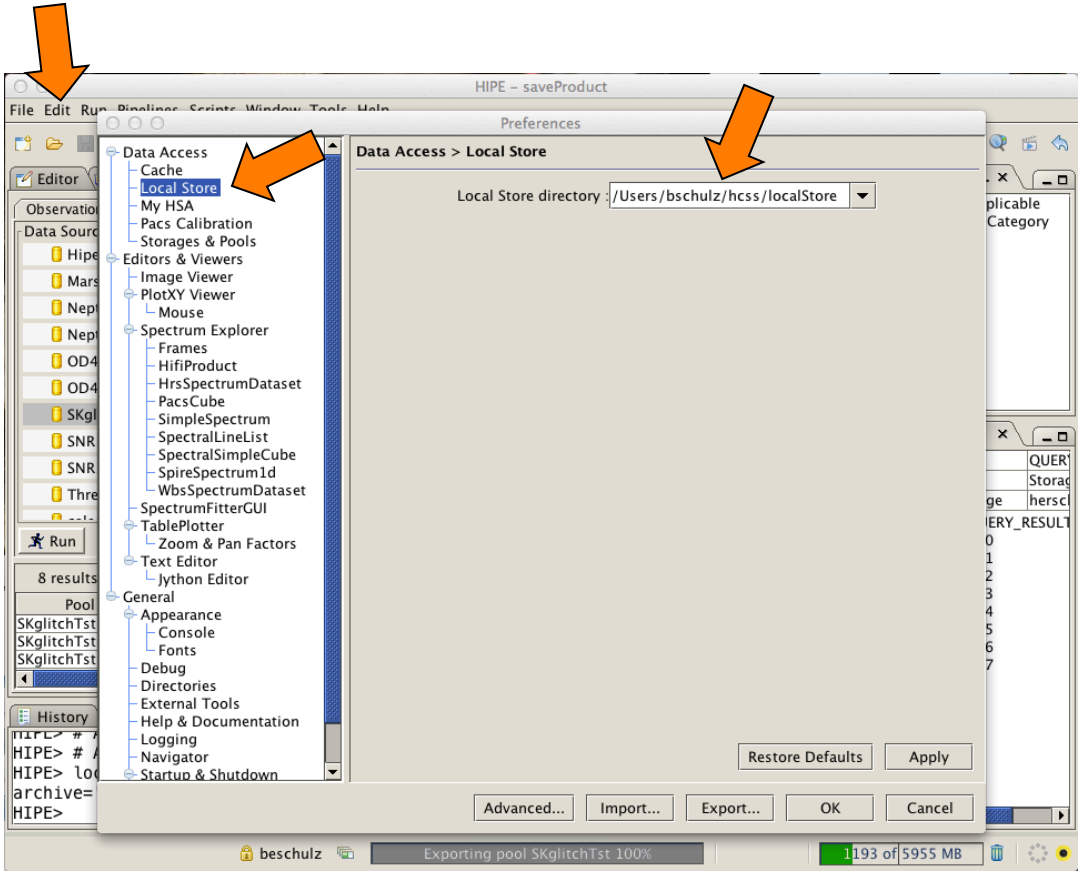
Managing Pools

- Select a pool and right-click.
- Pools can be created, renamed, moved to another storage directory, deleted, and exported.
- Moving will move the pool to another directory in your file system.
- Export will copy the contents of a pool into a zip file, so it can be sent to collaborators and other pundits.



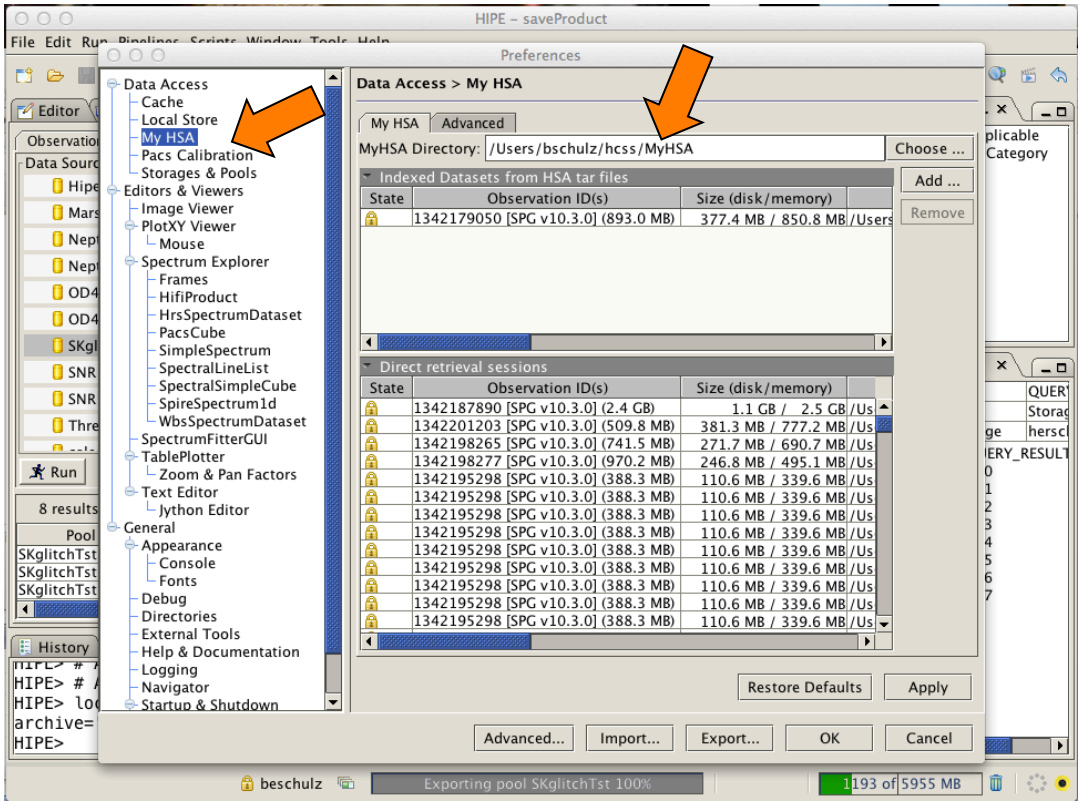
A Place for Ordinary Pools

- The location of the directory where the pools are stored can be determined in the "Preferences" menu.
- Select "Edit"→"Preferences".
- Select there "Data Access"→"Local Store".
- It makes sense to configure this to be on a large disk with sufficient space.



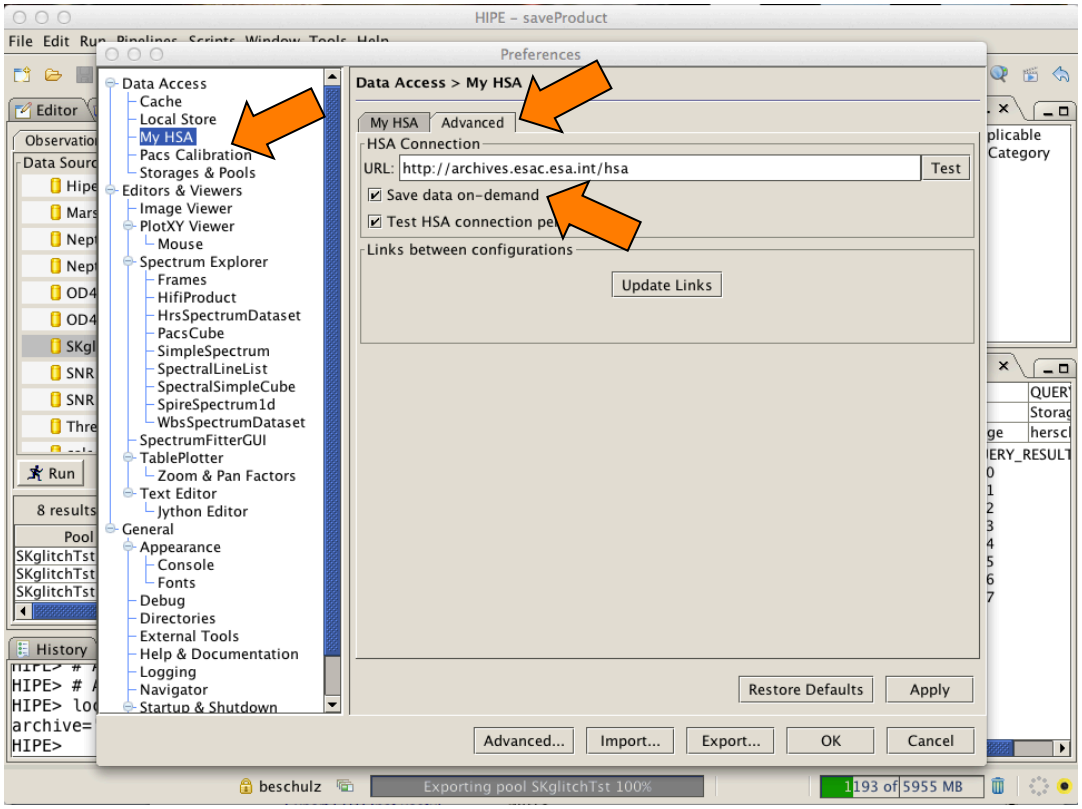
A Place for the MyHsa Pool

- The myHsa pool has a separate directory that can be configured also in “Preferences”
- Select “Edit”→”Preferences”.
- Select there “Data Access”→”My HSA”.
- It is good to choose a disk with sufficient space, when the “save” option in getObservation() is used or the save data on-demand option is activated in the “Advanced” tab.



On-Demand Save Option

- The save data on-demand option keeps copies of all products that are loaded through “lazy-loading” in the myHsa pool, making access to the same products faster next time.

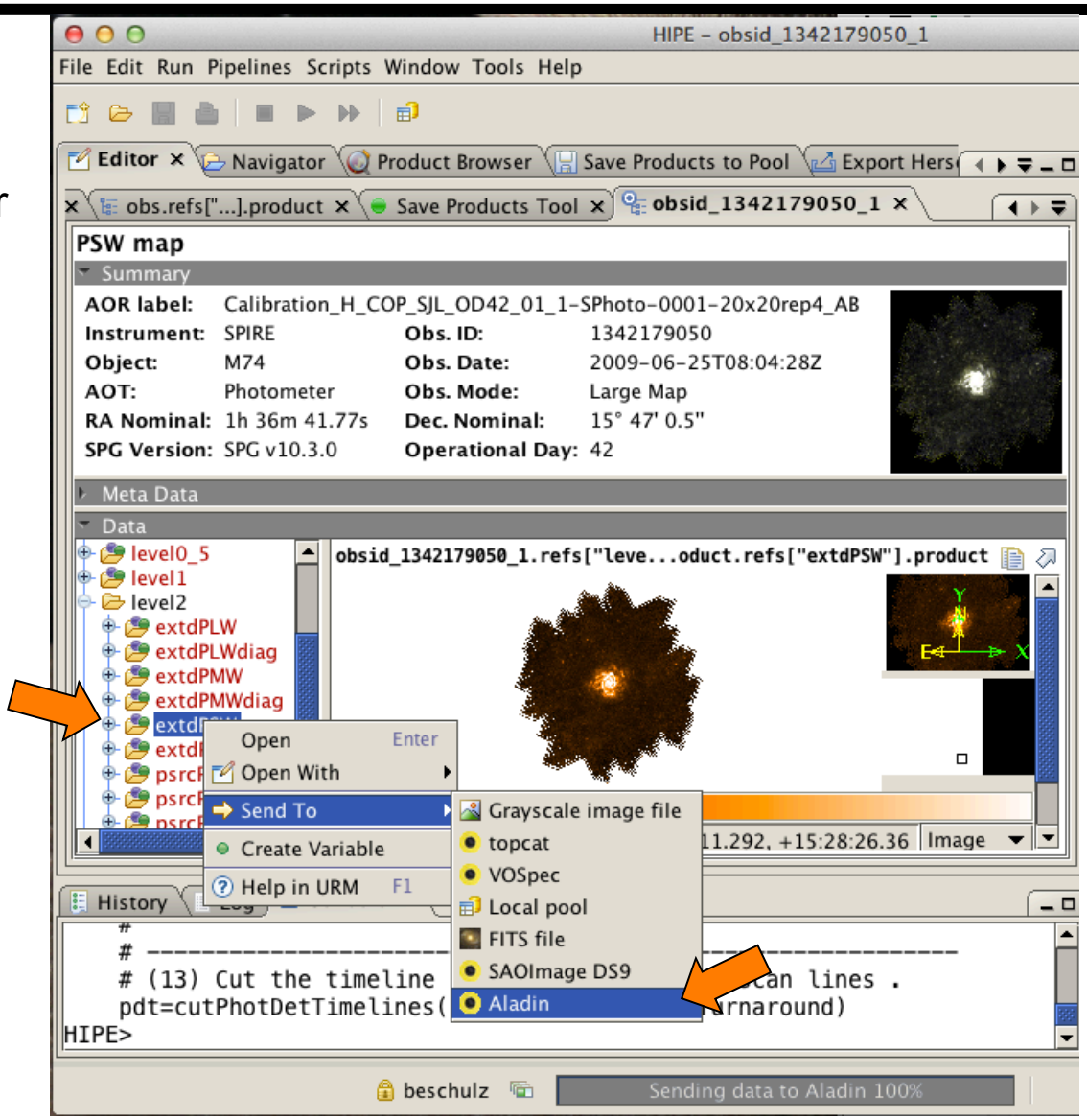


Save data on-demand is off by default

Data Exchange via SAMP (Images)

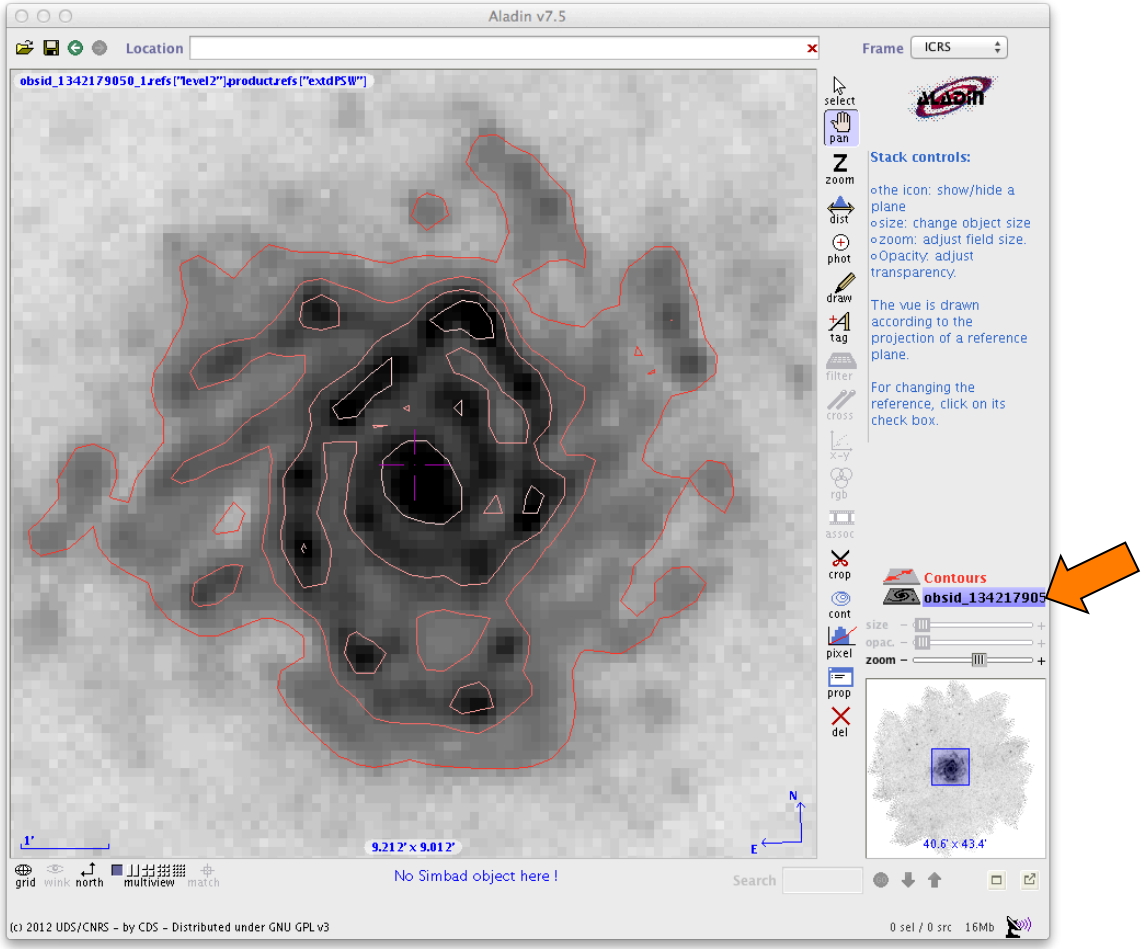
- HIPE provides support for a powerful interface to share data between applications, the Simple Application Messaging Protocol (SAMP)
- The send-to menu provides options to send image data or tabular data to Topcat, VOSpec, DS9, or Aladin.
- Right click on a map and send it to Aladin.
- The Java application is downloaded and opens the image.
- Many more visualization and analysis methods become available.

See: http://www.ivoa.net/documents/SAMP/



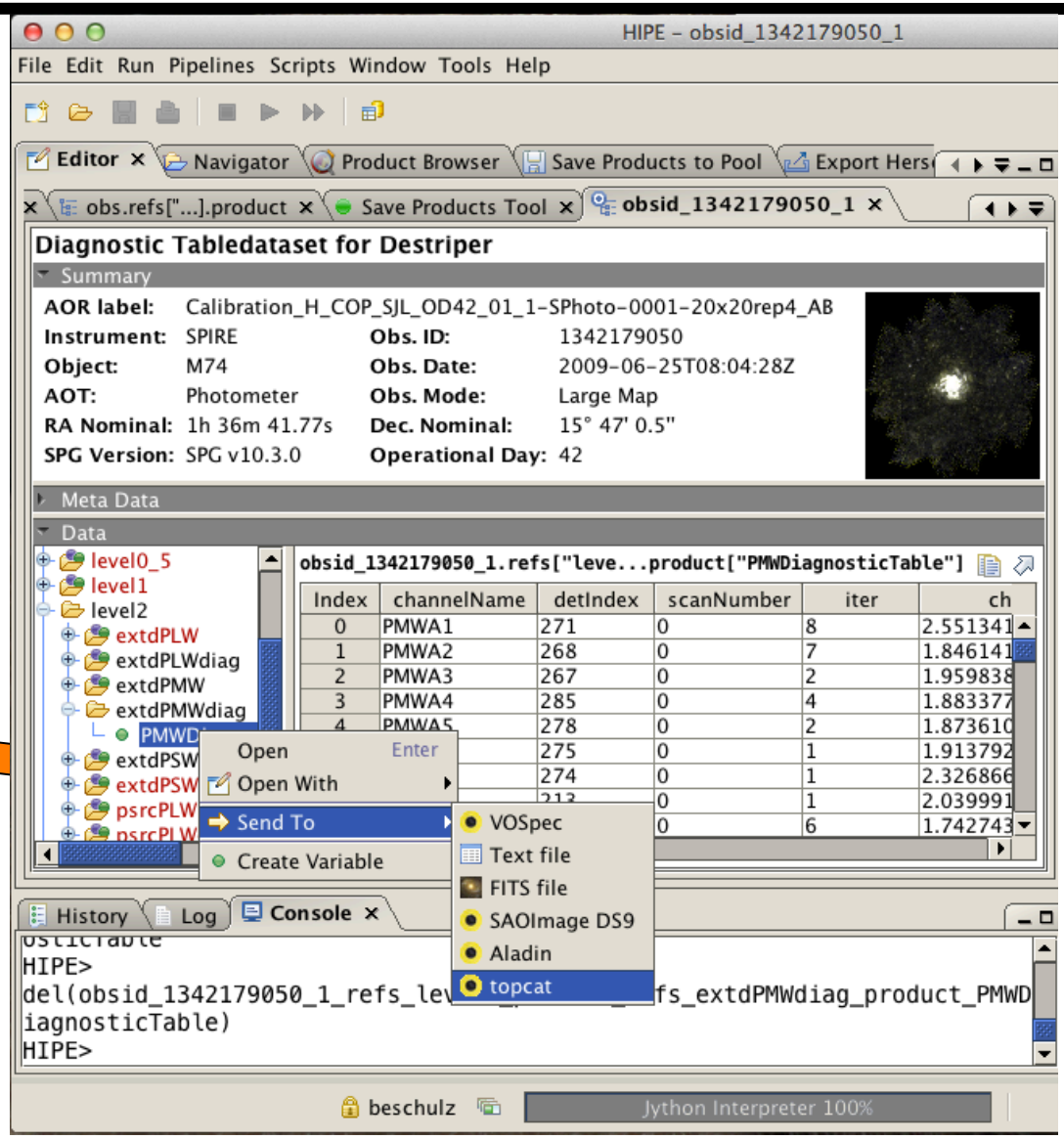
Data Exchange via SAMP (Images)

- HIPE provides support for a powerful interface to share data between applications, the Simple Application Messaging Protocol (SAMP)
- The send-to menu provides options to send image data or tabular data to Topcat, VOSpec, DS9, or Aladin.
- Right click on a map and send it to Aladin.
- The Java application is downloaded and opens the image.
- Many more visualization and analysis methods become available.



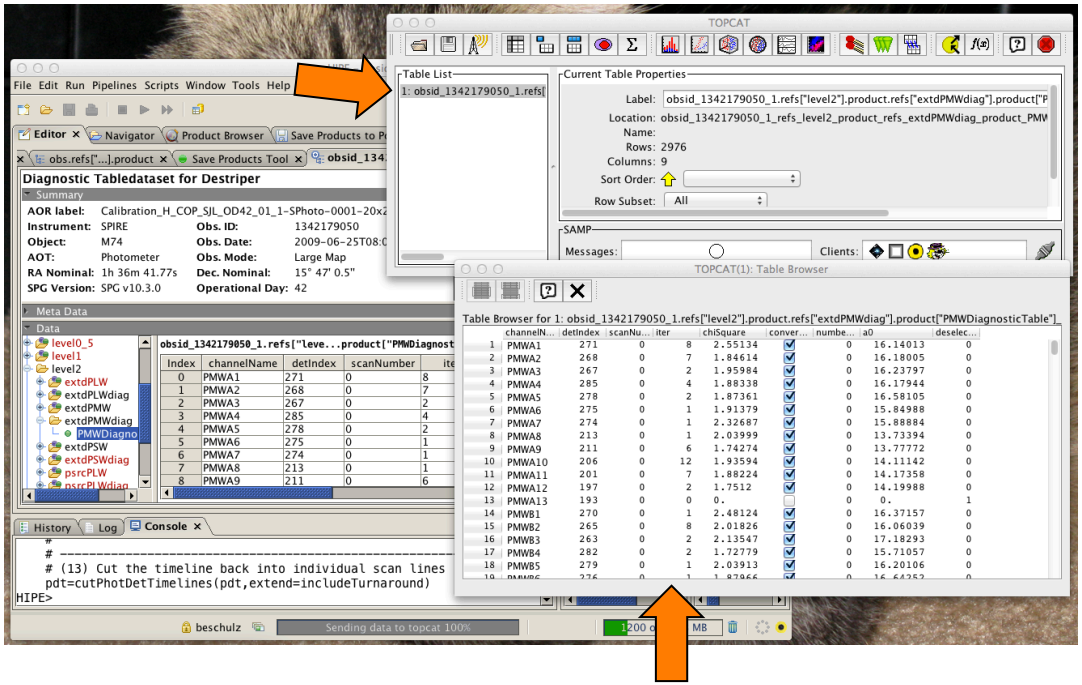
Data Exchange via SAMP (Tables)

- Another example sending a table dataset.
- Start Topcat on your computer.
- Select a SPIRE diagnostic table and send it to Topcat.
- It appears as a table and can be analyzed using the Topcat features.
- It can even be modified on Topcat and sent back to HIPE using the "Transmit table to all applications using SAMP" button.



Data Exchange via SAMP (Tables)

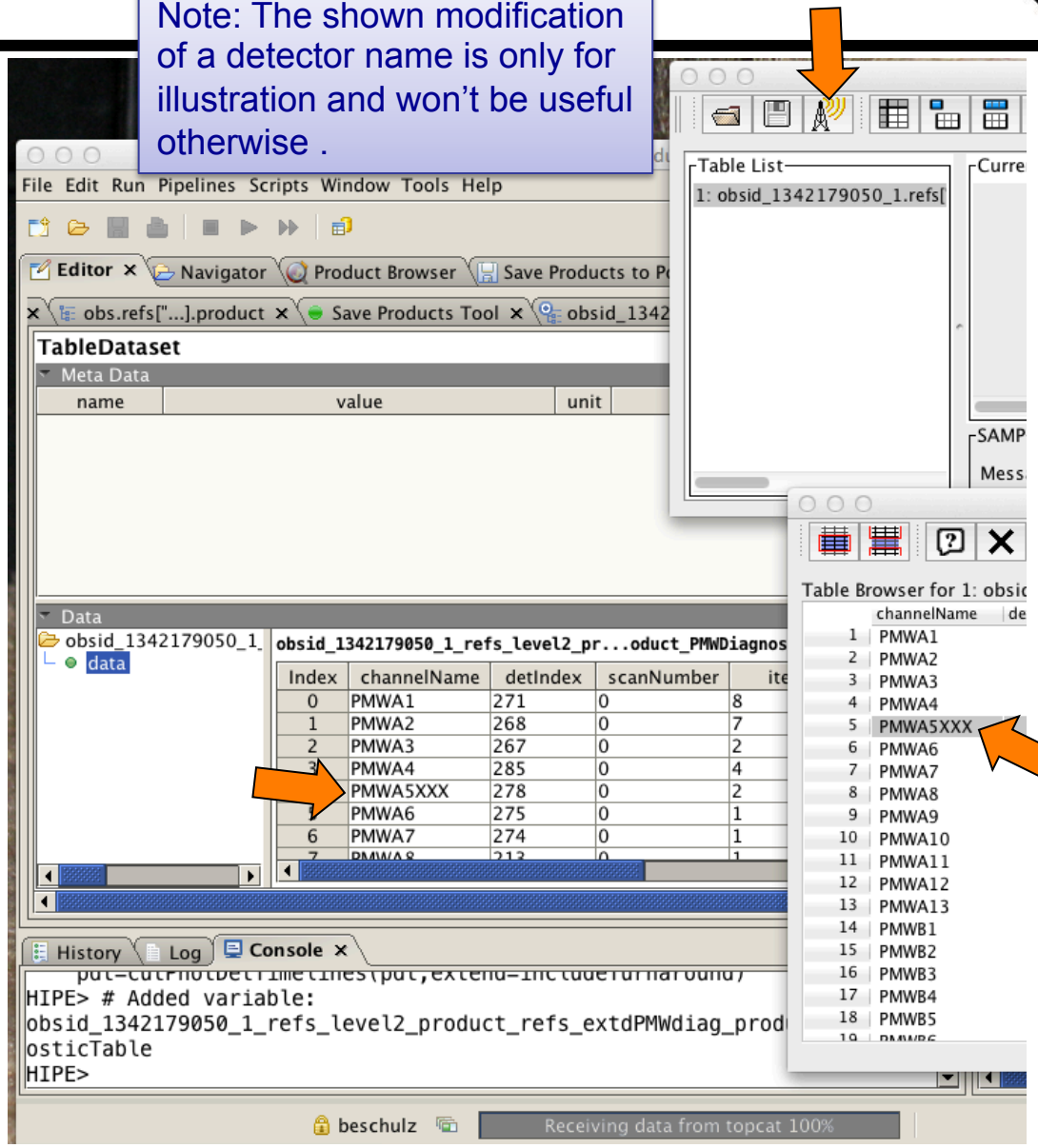
- Another example sending a table dataset.
- Start Topcat on your computer.
- Select a SPIRE diagnostic table and send it to Topcat.
- It appears as a table and can be analyzed using the Topcat features.
- It can even be modified on Topcat and sent back to HIPE using the “Transmit table to all applications using SAMP” button.



Data Exchange via SAMP (Tables)

- Another example sending a table dataset.
- Start Topcat on your computer.
- Select a SPIRE diagnostic table and send it to Topcat.
- It appears as a table and can be analyzed using the Topcat features.
- It can even be modified on Topcat and sent back to HIPE using the “Transmit table to all applications using SAMP” button.

Note: The shown modification of a detector name is only for illustration and won’t be useful otherwise .



Tasks

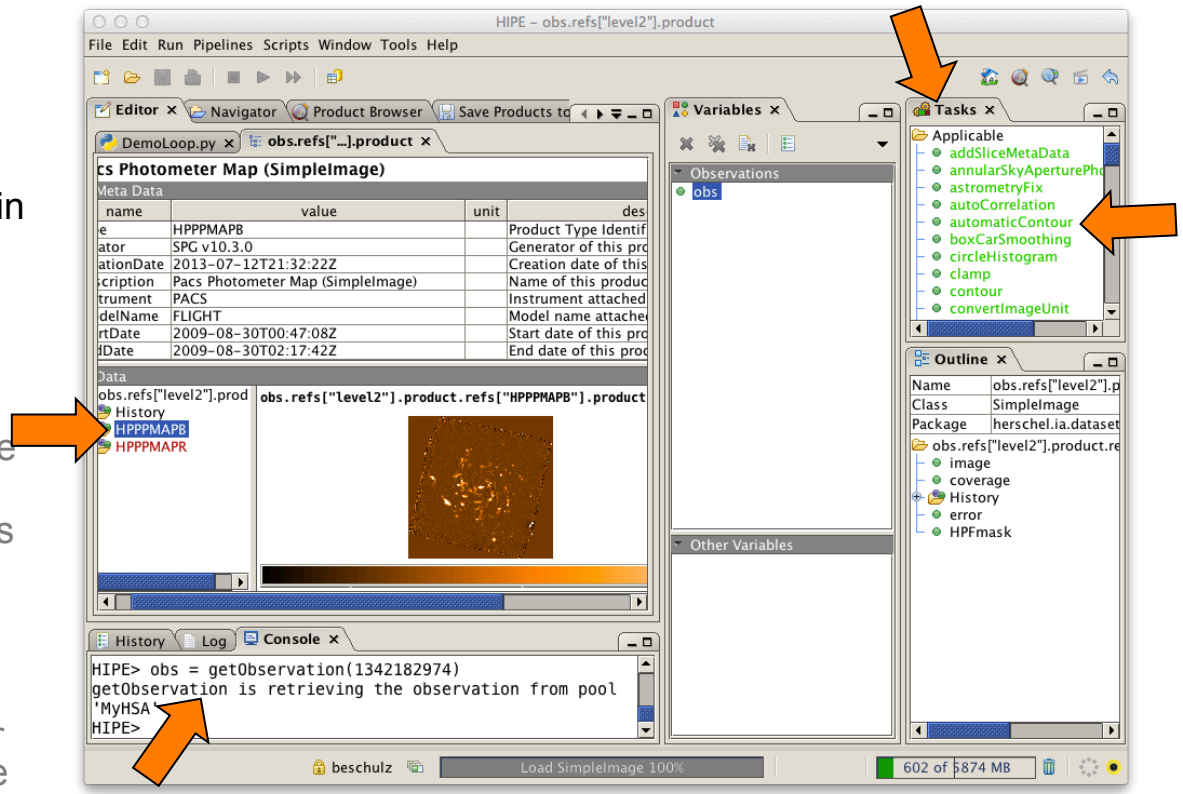
- Tasks are special classes in HIPE conforming to a rigorous specification of input and output parameters.

- They are registered with the system and appear in the Task View.
- Calling them from the Task View will produce a default GUI regardless of whether GUI features were specified in the code.
- They can be made applicable to specific product types and will appear in the "Applicable" folder only for these.
- There is a large collection of pre-defined tasks available in HIPE.

- Users can write their own tasks in Jython if needed and distribute them as a HIPE Plug-In.
- SPIA and CASSIS are examples for HIPE Plug-Ins.

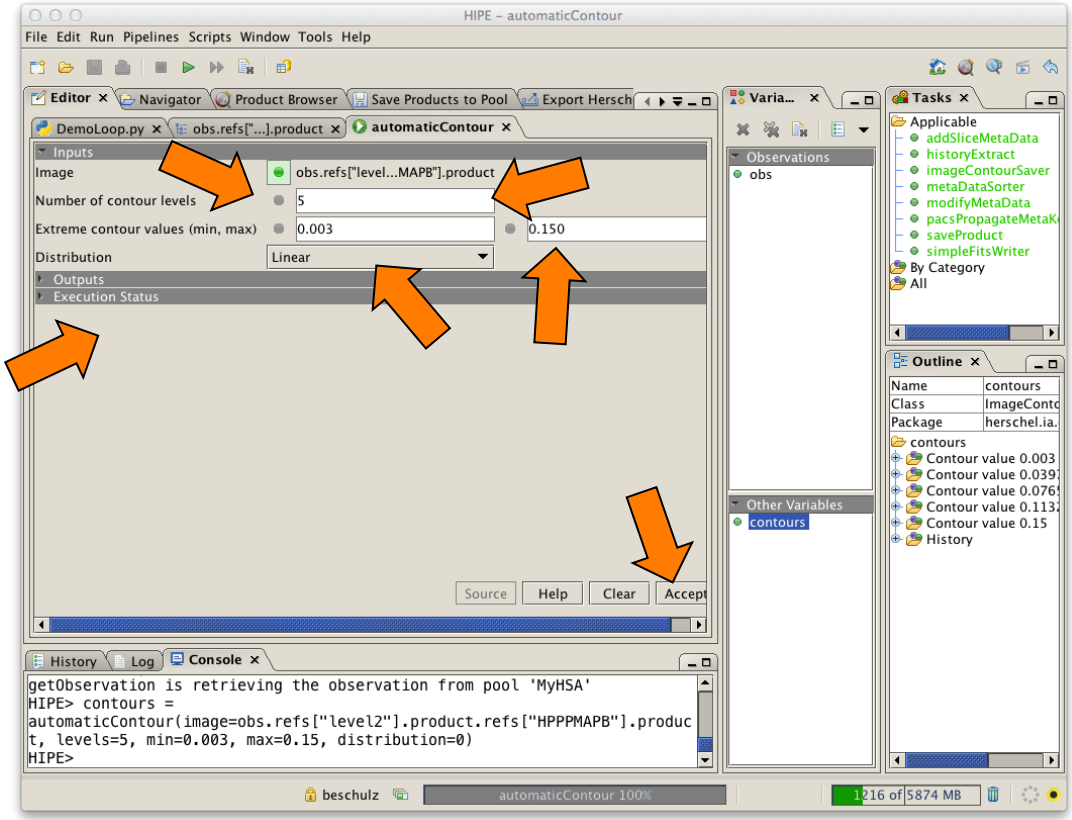
Task Example

- Tasks appear in the Tasks View on the upper right in this perspective.
- Load a PACS observation, click “Level 2’ in the Outline View and select HPPBMAPB in the appearing viewer.
- Double-click the “Applicable” folder in the Task View.
- Find and double-click the task “automaticContour”.
- The task GUI will appear in the Editor View.
- Enter some appropriate values and hit the “Accept” button.
- This will run the task.
- Note the script output in the Console View.
- Go back to the Level 2 Viewer and display the image with the Standard Image Viewer (right click).
- Drag the variable “contours” over the image.



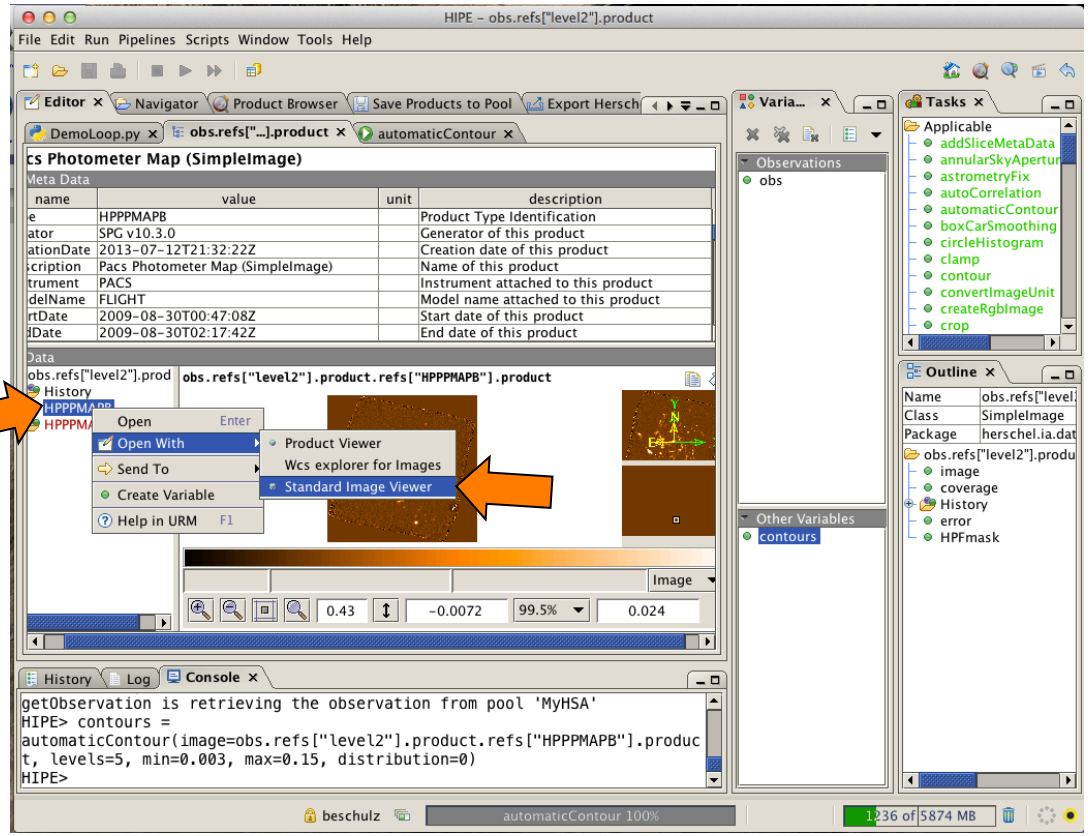
Task Example

- Tasks appear in the Tasks View on the upper right in this perspective.
- Load a PACS observation, click “Level 2’ in the Outline View and select HPPBMAPB in the appearing viewer.
- Double-click the “Applicable” folder in the Task View.
- Find and double-click the task “automaticContour”.
- The task GUI will appear in the Editor View.
- Enter some appropriate values and hit the “Accept” button.
- This will run the task.
- Note the script output in the Console View.
- Go back to the Level 2 Viewer and display the image with the Standard Image Viewer (right click).
- Drag the variable “contours” over the image.



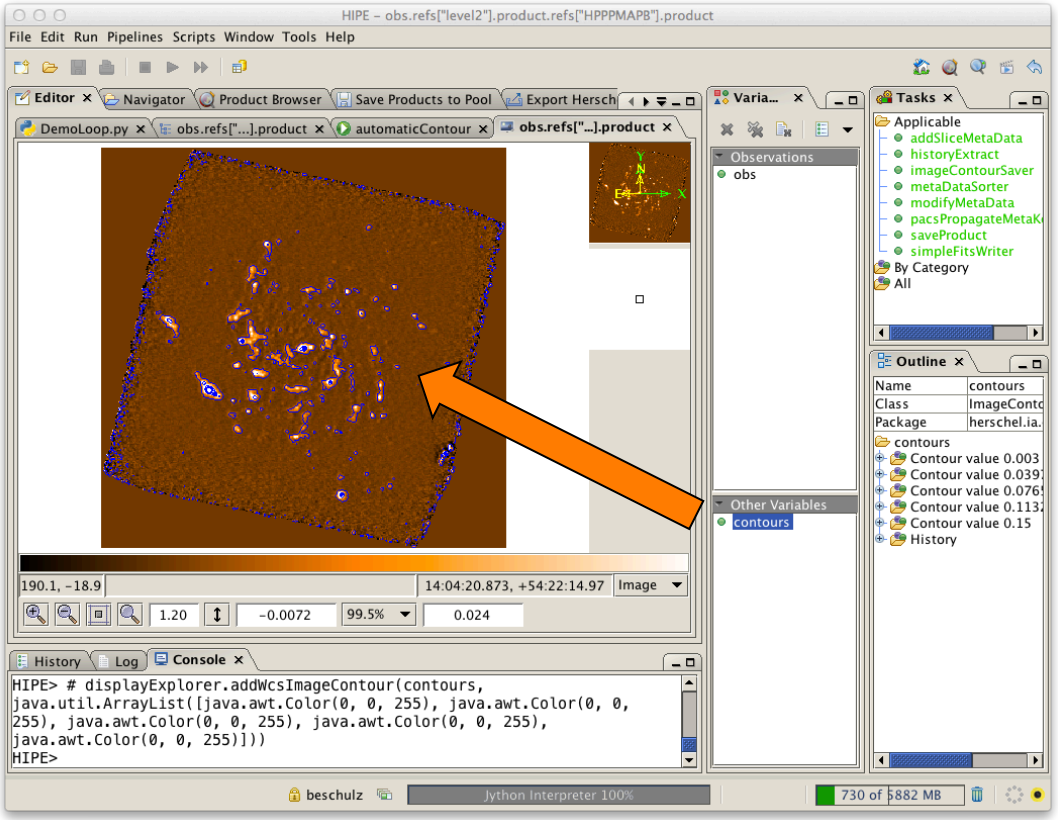
Task Example

- Tasks appear in the Tasks View on the upper right in this perspective.
- Load a PACS observation, click “Level 2’ in the Outline View and select HPPBMAPB in the appearing viewer.
- Double-click the “Applicable” folder in the Task View.
- Find and double-click the task “automaticContour”.
- The task GUI will appear in the Editor View.
- Enter some appropriate values and hit the “Accept” button.
- This will run the task.
- Note the script output in the Console View.
- Go back to the Level 2 Viewer and display the image with the Standard Image Viewer (right click).
- Drag the variable “contours” over the image.



Task Example

- Tasks appear in the Tasks View on the upper right in this perspective.
- Load a PACS observation, click “Level 2’ in the Outline View and select HPPBMAPB in the appearing viewer.
- Double-click the “Applicable” folder in the Task View.
- Find and double-click the task “automaticContour”.
- The task GUI will appear in the Editor View.
- Enter some appropriate values and hit the “Accept” button.
- This will run the task.
- Note the script output in the Console View.
- Go back to the Level 2 Viewer and display the image with the Standard Image Viewer (right click).
- Drag the variable “contours” over the image.



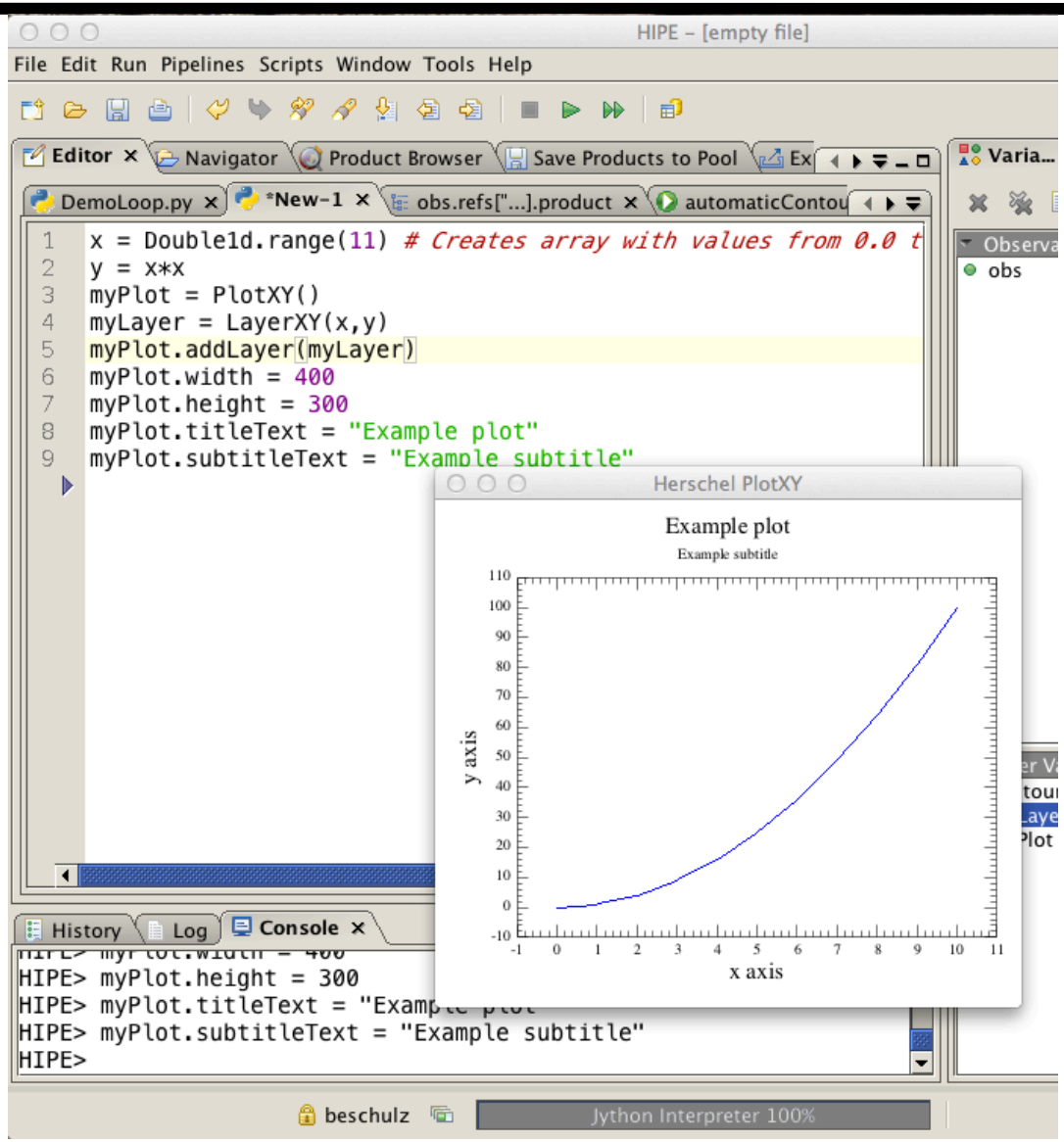
Many more tasks are available, in particular for map arithmetic, photometry, line fitting etc.

Libraries/Packages

- Hipe contains a number of software Libraries that are available to scripting.
- Examples are:
  - Numerics Library
  - Plot Library
  - Image toolbox
  - Spectral toolbox
  - Product Access Layer
  - etc…
- A good overview over available functionality is in the "Categorized view of commands" in the "HCSS User's Reference Manual"

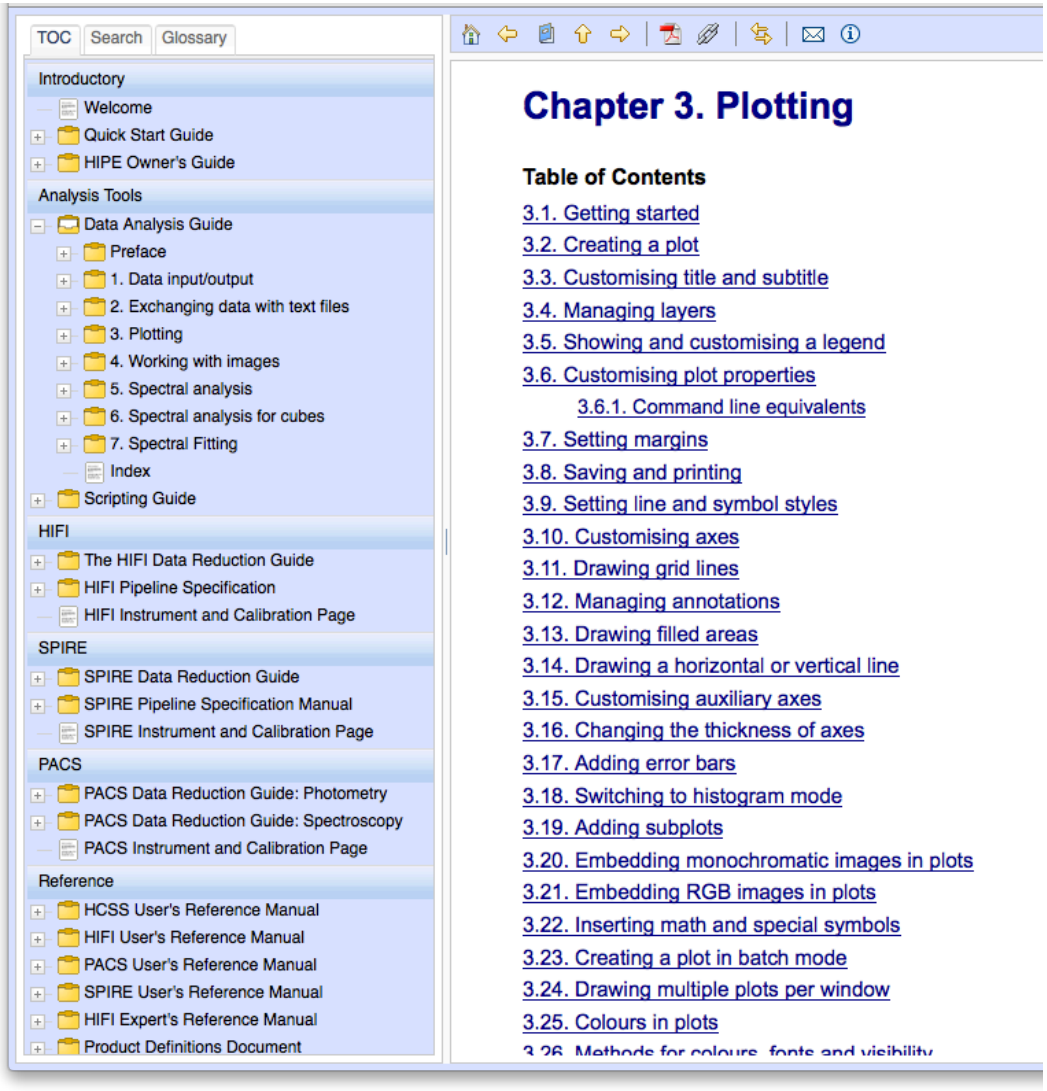
Plot Library

- HIPE contains a powerful plot library with many ways to create high quality publication ready diagrams.
- A simple example code is shown to the right and the resulting plot.
- For more information refer to the documentation in the Data Analysis Guide.



Plot Package

- HIPE contains a powerful plot library with many ways to create high quality publication ready diagrams.
- A simple example code is shown to the right and the resulting plot.
- For more information refer to the documentation in the Data Analysis Guide.



Some Practical Points at the End

- This was only the tip of the Iceberg. HIPE is a powerful data processing and analysis system with many more aspects to explore.

- Configuring HIPE is important at the start.
- Set the places for the local pools (localStore) and for the HSA cache pool (myHsa) to a disk with enough space.
- The amount of RAM to reserve for HIPE should be adjusted to the RAM available and should leave some room for the operating system.
- See the Preferences menu for other adjustments, like the appearance of the console. In many cases you will have to re-start the session, so configure first.
- Instruments will have additional configuration requirements.

- There is more documentation available than you might wish for but we are here to give you the necessary guidance.